ENKBD-2.0 VAX 11/730 8085 TEST PART IV

EP-ENKBD-DL-2.0 FICHE 1 OF 1

JAN 1983
COPYRIGHT© 82-83
MADE IN USA

dl9080

IDENTIFICATION

Product code: ENKBD VERSION 2.0

Product name: 8085 BASED MICRO-DIAGNOSTIC FOR VAX-11/730 WCS AND CPU MODULES

Product date: 11-OCT-82

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1982 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC	DECsystem-10	DECSYSTEM-20
DECUS	MASSBUS	PDP
UNIBUS	VAX	VMS

digital

Table of Contents

1.0 ABSTRACT 3

2.0 HARDWARE REQUIREMENTS 3

3.0 SOFTWARE REQUIREMENTS 3

4.0 PREREQUISITES 3

5.0 OPERATING INSTRUCTIONS 3
5.1 Options 3
5.2 Event Flags 4
5.3 APT Setup 4

6.0 PROGRAM FUNCTIONAL DESCRIPTION 4
6.1 Program Overview 4
6.2 Program Size 4
6.3 Program Run Times 4
6.4 Run-time Dynamics 4
6.5 Fault Detection 5
6.6 Performance During Hardware Failures 5
6.7 Program Applications 5
6.8 Test Descriptions 6

7.0 MAINTENANCE HISTORY 6

1.0 ABSTRACT

This program is an 8085 based micro-diagnostic designed to test the hardware on the WCS (Writable Control Store) and DAP (Data Path) modules of the VAX-11/730 processor.

2.0 HARDWARE REQUIREMENTS

This program will run with a minimum hardware configuration of the WCS and CPU modules present.

Other hardware modules and options may be installed without affecting the diagnostic's performance.

3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into the top 3K of 8085 RAM. The micro-diagnostic monitor also resides in 8085 ram, where console software is normally present.

4.0 PREREQUISITES

This module should be run after running modules ENKBA thru ENKBC.

5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKBD after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

5.1 Options

Not applicable

5.2 Event Flags

Not applicable

5.3 APT Setup

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port.

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

The following descibes the APT parameters needed to execute this diagnostic:
TBS

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. A bottom up diagnostic strategy is utilized which builds on previous tests. The tests should be run in consecutive order, to gain the best isolation of a failing component.

6.2 Program Size

This program runs in 8085 RAM memory and is approximately 3K bytes long.

6.3 Program Run Times

This program will take approximately 10 seconds to run.

6.4 Run-time Dynamics

Not applicable

6.5 Fault Detection

The diagnostic will report errors with the following header:

```
SECT  TST ERR   EXP   REC  OTHER  MSK  MODULE
```

1. SECT is the program name (ENKBD)
2. TST is the test number that failed.
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.6 Performance During Hardware Failures

A power fail will cause a rebooting of the system (if a battery backup is not present). Other hardware failures will cause normal error printouts.

6.7 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is APT compatable.

Customers may use the program to verify the basic integrity of the WCS and CPU modules in the system.

6.8 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
8-MAR-82	1.0	Initial release.
11-OCT-82	2.0	RD changes.

```
7000    3
7000    4   ;       ENKBD  8085 based diagnostic for WCS/DAP module  Rev 02.00
7000    5   ;       Table of Contents
7000    6   ;
7000    7   ;  134  DATA AREA
7000    8   ;  323  PROGRAM ENTRY POINT
7000    9   ;  342  VARIABLES AREA
7000   10   ;  397  PROGRAM SPECIFIC SUBROUTINES
7000   11   ;  398    CHECK_RD
7000   12   ;  417    MASTER_RESET
7000   13   ;  432    MASTER_RESET_RESTORE
7000   14   ;  446    CLR_CG_PNT
7000   15   ;  460    INC_CG_PNT
7000   16   ;  474    WRITE_MASTER
7000   17   ;  491    READ_MASTER
7000   18   ;  510    WRITE_MODE
7000   19   ;  530    READ_MODE
7000   20   ;  550    WRITE_LOAD
7000   21   ;  569    READ_LOAD
7000   22   ;  589    WRITE_HOLD
7000   23   ;  609    READ_HOLD
7000   24   ;  628    WRITE_ALARM
7000   25   ;  653    READ_ALARM
7000   26   ;  678    WRITE_ITDB
7000   27   ;  698    READ_ITDB
7000   28   ;  716    MM_BITS
7000   29   ;  737    INC_CYCLE_CNT
7000   30   ;  749    DISARM_CNTR
7000   31   ;  777    LOAD_CNTR
7000   32   ;  805    SAVE_CNTR
7000   33   ;  832    SET_OUTPUT
7000   34   ;  850    CLR_OUTPUT
7000   35   ;  865    INC_CNTR
7000   36   ;  882    LOAD_MOD_DAT
7000   37   ;  897    TEST1E_ERROR
7000   38   ;  927    TEST1F_ERROR
7000   39   ;  960    TEST21_ERROR
7000   40   ;  993    TEST22_ERROR
7000   41   ; 1023    TEST24_ERROR
7000   42   ; 1052    TEST26_ERROR
7000   43   ; 1087    TEST20_ONES_ZEROS
7000   44   ; 1047  INTERVAL_TIMER TESTS
7000   45   ; 1230    TEST 1E - MASTER RESET TEST
7000   46   ; 1397    TEST 1F - MASTER MODE REGISTER TEST
7000   47   ; 1571    TEST 20 - ELEMENT CYCLE REGISTER TEST
7000   48   ; 1790    TEST 21 - ALARM REGISTER TEST
7000   49   ; 1985    TEST 22 - DATA POINTER SEQUENCING TEST
7000   50   ; 2279    TEST 23 - COUNTER LOAD AND SAVE TEST
7000   51   ; 2439    TEST 24 - OUTPUT BIT SET/CLEAR TEST
7000   52   ; 2590    TEST 25 - COUNTER CARRY TEST
7000   53   ; 2704    TEST 26 - COMPARATOR TEST
7000   54
7000   55
7000   56
7000   57
7000   58
7000   59
7000   60
7000   61
7000   62
```

```
.TITLE  "ENKBD  8085 based diagnostic for WCS/DAP module  Rev 02.00"
;
;                          COPYRIGHT (c) 1982 BY
;                  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
;                   MASSACHUSETTS.  ALL RIGHTS RESERVED.
;
; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY  BE  USED  AND  COPIED
; ONLY  IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
; OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES THEREOF
; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER  PERSON.  NO
; TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
;
; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
; SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
;
; DIGITAL  ASSUMES  NO  RESPONSIBILITY  FOR  THE  USE OR RELIABILITY OF ITS
; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
;
;++
; FACILITY:     VAX-11/730 MICRO-DIAGNOSTIC.
;
; ABSTRACT:     This 8085 based diagnostic tests the hardcore logic of the
;               WCS and DAP modules.
;
; ENVIRONMENT:  ENKAA  Micro-diagnostic Monitor
;
; AUTHOR:       Jim Klumpp      4-MAY-81
;
; MODIFIED BY:  David Mayo      8-MAR-82        VERSION 01.00
;               Ernie Preisig   2-SEP-82        VERSION 02.00   RD changes
;--
```

```
7000  128
7000  129
7000  130
7000  131
7000  132
7000  133
7000  134 000                                  .PSECT  CPU4,BYTE
7000  135                                      .ALIGN  BYTE
7000  136
7000  137                                  ;*****************************************************************
7000  138                                  ;
7000  139                                  ; DATA AREA
7000  140                                  ;
7000  141                                  ;*****************************************************************
7000  142
7000  143 00000000                         HEAD            =0
7000  144
7000  145 00000000                                          EQUATE                  ;REGISTER EQUATE MACRO
          00000002
          00000004
          00000006
          00000006
          00000001
          00000003
          00000005
          00000007 00000006
7000  146
7000  147 00000000                         UPC14A          =<^X00>                 ;(RO) UPC BIT 14 ASSERTED HIGH <MSB>
7000  148
7000  149
7000  150 00000001                         MISC2           =<^X01>
7000  151 00000001                         BOOTF           =<^X01>                 ;(RO) BOOT FLAG. ASSERTED LOW. <LSB>
7000  152
7000  153
7000  154 00000002                         DCLO            =<^X02>                 ;(RO) DC LOW FLAG ASSERTED HIGH<MSB>
7000  155
7000  156 00000002                         HLTFLG          =<^X02>                 ;(RO) HALT FLAG ASSERTED LOW   <LSB>
7000  157
7000  158
7000  159 00000003                         READJ1          =<^X03>                 ;(RO) JUMPER J1 FLAG ASS. HIGH <MSB>
7000  160
7000  161 00000003                         ALLOWP          =<^X03>                 ;(RO) ENABLE CONTROL P ASS LOW <LSB>
7000  162
7000  163
7000  164 00000004                         READJ2          =<^X04>                 ;(RO) BIT 07 EQUALS J2 <MSB>
7000  165 00000004                         APTFLG          =<^X04>                 ;(RO) APT PRESENT FLAG <LSB>
7000  166
7000  167 00000005                         AUTO            =<^X05>                 ;(RO) AUTO TEST FLAG ASS LOW <LSB>
7000  168
7000  169 00000006                         REMDIS          =<^X06>                 ;(RO) REMOTE DISABLE FLAG ASS LOW <LSB>
7000  170
7000  171 00000007                         OKV15           =<^X07>                 ;(RO) 15 VOLTS OK FLAG ASS LOW <MSB>
7000  172
7000  173 00000007                         BOOTEN          =<^X07>                 ;(RO) BOOT ENABLE FLAG ASS LOW <LSB>
7000  174
7000  175 00000008                         WCSB0           =<^X08>                 ;(WO) WCS WRITE DATA REGISTER
7000  176 00000009                         WCSB1           =<^X09>
7000  177 0000000A                         WCSB2           =<^X0A>
7000  178
7000  179 0000001C                         ITDB            =<^X1C>                 ;(R/W) TIMER DATA REGISTER
```

```
7000  180 0000001D          ITCSR      =<^X1D>     ;(R/W) TIMER STATUS REGISTER
7000  181
7000  182 00000020          CLRCLK     =<^X20>     ;(WO) CLEAR CPU CLOCK RUN
7000  183 00000021          SETCLK     =<^X21>     ;(WO) SET CPU CLOCK RUN
7000  184
7000  185 00000022          CLRSS      =<^X22>     ;(WO) CLEAR CLOCK SINGLE STEP
7000  186 00000023          SETSS      =<^X23>     ;(WO) SET CLOCK SINGLE STEP
7000  187
7000  188 00000024          CLRCSK     =<^X24>     ;(WO) CLEAR CSR CLOCK
7000  189 00000025          SETCSK     =<^X25>     ;(WO) SET CSR CLOCK
7000  190
7000  191 00000026          CLRUPK     =<^X26>     ;(WO) CLEAR THE UPC CLOCK
7000  192 00000027          SETUPK     =<^X27>     ;(WO) SET THE UPC CLOCK
7000  193
7000  194 00000028          SETMCI     =<^X28>     ;(WO) SET MEMORY CONTROL INIT
7000  195 00000029          CLRMCI     =<^X29>     ;(WO) CLEAR MEMORY CONTROL INIT
7000  196
7000  197 0000002A          SETMCK     =<^X2A>     ;(WO) SET MEMORY CONTROL CLOCK
7000  198 0000002B          CLRMCK     =<^X2B>     ;(WO) CLEAR MEMORY CONTROL CLOCK
7000  199
7000  200 0000002C          CLRTMR     =<^X2C>     ;(WO) STOP TIMER
7000  201 0000002D          SETTMR     =<^X2D>     ;(WO) START TIMER COUNTING
7000  202
7000  203 0000002E          CLRBSY     =<^X2E>     ;(WO) CLEAR UNIBUS BUS BUSY
7000  204 0000002F          SETBSY     =<^X2F>     ;(WO) SET UNIBUS BUS BUSY
7000  205
7000  206 00000030          SETDCL     =<^X30>     ;(WO) SET DC LOW
7000  207 00000031          CLRDCL     =<^X31>     ;(WO) CLEAR DC LOW
7000  208
7000  209 00000032          SETACL     =<^X32>     ;(WO) SET AC LOW
7000  210 00000033          CLRACL     =<^X33>     ;(WO) CLEAR AC LOW
7000  211
7000  212 00000034          INITL      =<^X34>     ;(WO) CLEAR UNIBUS INIT
7000  213 00000035          INITH      =<^X35>     ;(WO) SET UNIBUS INIT
7000  214
7000  215 00000036          CLRWCK     =<^X36>     ;(WO) CLEAR WCS BIT 00
7000  216 00000037          SETWCK     =<^X37>     ;(WO) SET WCS BIT 00
7000  217
7000  218 00000038          CLRCSR     =<^X38>     ;(WO) CLEAR WCS BIT 01
7000  219 00000039          SETCSR     =<^X39>     ;(WO) SET WCS BIT 01
7000  220
7000  221 0000003C          CLRLIT     =<^X3C>     ;(WO) CLEAR THE RUN LIGHT
7000  222 0000003D          SETLIT     =<^X3D>     ;(WO) SET THE RUN LIGHT
7000  223
7000  224 00000044          TUDB       =<^X44>     ;(R/W) TU-58 DATA BUFFER
7000  225 00000045          TUSR       =<^X45>     ;(RO)  TU-58 STATUS REGISTER
7000  226 00000046          TUMODE     =<^X46>     ;(R/W) TU-58 MODE REGISTER 1 + 2
7000  227                                           ;FIRST ACCESS GET MR1,SECOND GETS MR2
7000  228                                           ;READING THE CR FORCES THE INTERNAL
7000  229                                           ;POINTER BACK TO MR1
7000  230 00000047          TUCR       =<^X47>     ;(R/W) TU-58 COMMAND REGISTER
7000  231
7000  232 00000048          TTDB       =<^X48>     ;(R/W) TERMINAL DATA BUFFER
7000  233 00000049          TTSR       =<^X49>     ;(RO)  TERMINAL STATUS REGISTER
7000  234 0000004A          TTMODE     =<^X4A>     ;(R/W) TERMINAL MODE REGISTER 1 + 2
7000  235                                           ;FIRST ACCESS GET MR1,SECOND GETS MR2
7000  236                                           ;READING THE CR FORCES THE INTERNAL
7000  237                                           ;POINTER BACK TO MR1
7000  238 0000004B          TTCR       =<^X4B>     ;(R/W) TERMINAL COMMAND REGISTER
7000  239
```

```
7000  240 00000080                    READ          =<^X80>          ;(RO) CONSOLE READ REGISTER
7000  241
7000  242 00000082                    CPUACK        =<^X82>          ;(RO) ACKNOWLEDGE FROM CPU       <LSB>
7000  243
7000  244 00000083                    CPATTN        =<^X83>          ;(RO) ATTENTION FROM CPU         <LSB>
7000  245
7000  246 00000084                    UPC14         =<^X84>          ;(RO) UPC BIT 14                 <LSB>
7000  247
7000  248 00000085                    CSR07         =<^X85>          ;(RO) CSR BIT 7                  <LSB>
7000  249
7000  250 00000086                    CSR15         =<^X86>          ;(RO) CSR BIT 15                 <LSB>
7000  251
7000  252 00000087                    CSR23         =<^X87>          ;(RO) CSR BIT 23                 <LSB>
7000  253
7000  254 000000A0                    ENBPAR        =<^XA0>          ;(WO) ENABLE STALL ON PARITY ERROR
7000  255 000000A1                    DISPAR        =<^XA1>          ;(WO) DISABLE STALL ON PARITY ERROR
7000  256
7000  257 000000A2                    VSHIFT        =<^XA2>          ;(WO) SET THE V-BUS TO SHIFT MODE
700   258 000000A3                    VNORML        =<^XA3>          ;(WO) SET THE V-BUS TO NORMAL MODE
7000  259
7000  260 000000A4                    SETTIM        =<^XA4>          ;(WO) SET THE INTERVAL TIMER INTERRUPT
7000  261 000000A5                    CLRTIM        =<^XA5>          ;(WO) CLEAR THE INTERVAL TIMER INTRPT.
7000  262
7000  263 000000A6                    ENBMEM        =<^XA6>          ;(WO) ENABLE MEMORY REFERENCES
7000  264 000000A7                    DISMEM        =<^XA7>          ;(WO) DISABLE MEMORY REFERENCES
7000  265
7000  266 000000A8                    SETACK        =<^XA8>          ;(WO) SET CONSOLE ACKNOWLEDGE
7000  267 000000A9                    CLRACK        =<^XA9>          ;(WO) CLEAR CONSOLE ACKNOWLEDGE
7000  268
7000  269                             ;NOTE:  THE CONSOLE ACKNOWLEDGE SIGNAL IS ALSO USED AS THE UPC
7000  270                             ;SERIAL INPUT.  WHEN USED THIS WAY THE SENSE IS INVERTED.  TO SIMPLIFY
7000  271                             ;MATTERS THE NEXT TWO EQUATES ARE INCLUDED
7000  272
7000  273 000000A8                    CLRUPC        =<^XA8>          ;(WO) CLEAR THE V-BUS SERIAL INPUT
7000  274 000000A9                    SETUPC        =<^XA9>          ;(WO) SET THE V-BUS SERIAL INPUT
7000  275
7000  276 000000AA                    SETATN        =<^XAA>          ;(WO) SET CONSOLE ATTENTION
7000  277 000000AB                    CLRATN        =<^XAB>          ;(WO) CLEAR CONSOLE ATTENTION
7000  278
7000  279 000000AC                    SETPFI        =<^XAC>          ;(WO) SET POWER FAIL INTERRUPT
7000  280 000000AD                    CLRPFI        =<^XAD>          ;(WO) CLEAR POWER FAIL INTERRUPT
7000  281
7000  282 000000AE                    SETHLT        =<^XAE>          ;(WO) SET THE CONSOLE HALT BIT
7000  283 000000AF                    CLRHLT        =<^XAF>          ;(WO) CLEAR THE CONSOLE HALT BIT
7000  284
7000  285 000000CC                    WRITE         =<^XCC>          ;(WO) THE CONSOLE WRITE REGISTER
7000  286
7000  287 000000EC                    YBUSRD        =<^XEC>          ;(RO) Y-BUS READ
7000  288
7000  289
7000  290
7000  291                             ;INTERVAL TIMER DATA           ;DATA TO...
7000  292
7000  293 000000FF                    RESET_DATA    = <^XFF>         ;EXECUTE A MASTER RESET.
7000  294 00000017                    MASTER_MODE   = <^X17>         ;POINT TO THE MASTER MODE REG.
7000  295 00000000                    MODE_REG      = <^X00>         ;POINT TO A COUNTER MODE REG.
7000  296 00000008                    LOAD_REG      = <^X08>         ;POINT TO A COUNTER LOAD REG.
7000  297 00000010                    HOLD_REG      = <^X10>         ;POINT TO A COUNTER HOLD REG.
7000  298 00000007                    ALARM_REG1    = <^X07>         ;POINT TO ALARM REG 1.
7000  299 0000000F                    ALARM_REG2    = <^X0F>         ;POINT TO ALARM REG 2.
```

```
7000  300 00000001          CG1_MODE_E       = <^X01>          ;POINT TO CG1 MODE REG IN ELEM CYC.
7000  301 00000019          CG1_HOLD_H       = <^X19>          ;POINT TO CG1 HOLD REG IN HOLD CYC.
7000  302 000000E8          DIS_DP_SEQ       = <^XE8>          ;DISABLE DATA POINTER SEQUENCING.
7000  303 000000C0          DISARM_CNTR_DAT  = <^XC0>          ;DISARM A COUNTER.
7000  304 00000040          LOAD_CNTR_DAT    = <^X40>          ;LOAD A COUNTER.
7000  305 000000A0          SAVE_CNTR_DAT    = <^XA0>          ;SAVE A COUNTER'S CONTENTS.
7000  306 000000E8          SET_OUTPUT_DAT   = <^XE8>          ;ASSERT A COUNTER'S OUTPUT.
7000  307 000000E0          CLR_OUTPUT_DAT   = <^XE0>          ;CLEAR A COUNTER'S OUTPUT.
7000  308 000000F0          INC_CNTR_DAT     = <^XF0>          ;SINGLE STEP A COUNTER.
7000  309
7000  310
7000  311                   ;ADDITIONAL DATA
7000  312
7000  313 00004C22          PARITY_VECTOR    = <^X4C22>        ;PARITY ERROR BRANCH VECTOR.
7000  314 00004C25          POWER_VECTOR     = <^X4C25>        ;POWER FAIL BRANCH VECTOR.
7000  315 000040D9          MIPFLG           = <^X40D9>        ;ADDRESS OF MODEM IN PROGRESS FLAG
7000  316
7000  317
7000  318                   ;*****************************************************************************
7000  319                   ;
7000  320                   ; PROGRAM ENTRY POINT.... MUST BE THE FIRST INSTUCTION IN THE PROGRAM
7000  321                   ;
7000  322                   ;*****************************************************************************
7000  323
7000  324 031A'C3           ENTRY:           JMP     TEST1E
7003  325
7003  326                   ;*****************************************************************************
7003  327                   ;
7003  328                   ; VERSION NUMBER AND LAST TWO LETTERS OF FILE NAME.  THESE 2 WORDS CAN NOT
7003  329                   ; BE MOVED TO ANY OTHER LOCATION WITHOUT A MICMON CHANGE.
7003  330                   ;
7003  331                   ;*****************************************************************************
7003  332
7003  333 0200              VERSION:         .WORD           <^X0200>        ;VERSION NUMBER (REV 02.00)
7005  334 44 42             FILE_NAME:       .ASCII          "BD"            ;LAST 2 CHARS OF FILE NAME
7007  335
7007  336                   ;*****************************************************************************
7007  337                   ;
7007  338                   ; VARIABLES AREA
7007  339                   ;
7007  340                   ;*****************************************************************************
7007  341
7007  342
7007  343 00                CG_PNT:          .BYTE           0               ;POINTS TO COUNTER GROUP.
7008  344 00                SHIFT_CNT:       .BYTE           0               ;COUNTS ACCUMUALTOR SHIFTS.
7009  345 0000              TIMER_DAT:       .WORD           0               ;DATA TO BE WRITTEN TO TIMER.
700B  346 00                CYCLE_CNT:       .BYTE           0               ;KEEPS TRACK OF POSITION IN
700C  347                                                                    ; ELEMENT, HOLD AND CONTROL
700C  348                                                                    ; GROUP CYCLES.
700C  349
700C  350 000A              NORMAL_MASTER:   .WORD           <^X000A>        ;DATA FOR NORMAL MASTER REG
700E  351                                                                    ; EXPECTED BY BOOT CODE
700E  352 00                ZERO_DAT:        .BYTE           <^X00>          ;*
700F  353 00                ONE_SHIFT:       .BYTE           <^X00>          ;*
7010  354 01                                 .BYTE           <^X01>          ;STANDARD DATA PATTERNS.
7011  355 FF                ONE_DAT:         .BYTE           <^XFF>          ;*
7012  356 FF                ZERO_SHIFT:      .BYTE           <^XFF>          ;*
7013  357 FE                                 .BYTE           <^XFE>          ;*
7014  358
7014  359 000B              TEST1E_DAT:      .WORD           <^X000B>        ;EXPECTED COUNTER MODE DATA
```

```
7016                                                                    ; AFTER MASTER RESET.
7016
7016  5555    TEST22_DAT:   .WORD    <^X5555>    ;ALTERNATING PATTERN OF ONES
7018                                             ; AND ZEROS.
7018
7018  000A    TEST23_DAT:   .WORD    <^X000A>    ;MASTER MODE DATA.
701A  2800                  .WORD    <^X2800>    ;COUNTER MODE DATA.
701C
701C  0200    TEST24_DAT:   .WORD    <^X0200>    ;COUNTER MODE DATA.
701E
701E  0800    TEST25_DAT:   .WORD    <^X0800>    ;COUNTER MODE DATA.
7020
7020  0C0A    TEST26_DAT:   .WORD    <^X0C0A>    ;MASTER MODE REGISTER DATA.
7022  0800                  .WORD    <^X0800>    ;COUNTER MODE REGISTER DATA.
7024
7024
7024          ;*****************************************************************
7024          ;
7024          ;  SHIFT VARIABLES - DO NOT SEPARATE PAIRS
7024          ;
7024          ;*****************************************************************
7024
7024
7024  02      TEMP_SHIFT:   .BYTE  2          ;TEMPORARY STORAGE LOCATION FOR
7025  00000027 TEMP_DAT:    .BLKB  2          ; DATA FROM VARIOUS TESTS.
7027
7027  02      EXP_SHIFT:    .BYTE  2          ;LOCATION OF EXPECTED DATA FROM
7028  0000002A EXP_DAT:     .BLKB  2          ; VARIOUS TESTS.
702A
702A  02      REC_SHIFT:    .BYTE  2          ;LOCATION OF RETURNED DATA FROM
702B  0000002D REC_DAT:     .BLKB  2          ; VARIOUS TESTS.
702D
702D          ;*****************************************************************
702D          ;
702D          ;  CHECK_RD     THIS ROUTINE CHECKS TO SEE IF DIAGNOSTIC IS UNDER RD CONTROL.
702D          ;               IF IT IS, A CALL IS MADE TO GCVECT SO THAT THE BOOT CODE CAN
702D          ;               KEEP TRACK OF ELAPSED TIME.
702D          ;*****************************************************************
702D
702D  40D9 3A CHECK_RD:     LDA    MIPFLG       ;GET STATE OF MODEM IN PROGRESS
7030  B7                    ORA    A            ;SET CONDITION CODES
7031  C8                    RZ                  ;RETURN IF ZERO
7032  0000'CD               CALL   GCVECT       ;ELSE GO TO BOOT CODE IF RD
7035
7035  0000'3A               LDA    CNTLC_TYPED  ;SEE IF CONTROL C TYPED (UNDER RD)
7038  0F                    RRC
7039  0000'DC               CC     SET_FOR_CONT ;IF SO, GO TO MONITOR.
703C
703C  C9                    RET                 ;BACK TO TEST
703D
703D
703D          ;*****************************************************************
703D          ;
703D          ;  MASTER_RESET  THIS ROUTINE PERFORMS AN INTERVAL TIMER MASTER RESET.
703D          ;
703D          ;*****************************************************************
703D
703D
703D  FF 3E   MASTER_RESET: MVI    A,RESET_DATA ;MOVE MASTER RESET DATA TO A.
703F  1D D3                 OUT    ITCSR        ;PERFORM MASTER RESET.
```

```
7041                                    RET

7042                    ;*****************************************************************
7042                    ;
7042                    ;  MASTER_RESET_RESTORE  THIS ROUTINE PERFORMS AN INTERVAL TIMER MASTER RESET.
7042                    ;                 IT THEN WRITES THE MASTER MODE REGISTER TO 0A00 WHICH IS THE
7042                    ;                 VALUE EXPECTED TO BE THERE IN THE BOOT BLOCK.
7042                    ;
7042                    ;*****************************************************************

7042                    MASTER_RESET_RESTORE:
7042 FF 3E                              MVI     A,RESET_DATA    ;MOVE MASTER RESET DATA TO A.
7044 1D D3                              OUT     ITCSR           ;PERFORM MASTER RESET.
7046
7046 00'0C'21                           LXI     H,NORMAL_MASTER ;POINT TO 0A00 DATA
7049 0057'CD                            CALL    WRITE_MASTER    ;WRITE NORMAL MASTER REG DATA
704C
704C C9                                 RET

704D                    ;*****************************************************************
704D                    ;
704D                    ;  CLR_CG_PNT  THIS ROUTINE CLEARS THE COUNTER GROUP POINTER.
704D                    ;
704D                    ;*****************************************************************

704D AF                 CLR_CG_PNT:     XRA     A               ;CLEAR A.
704E 0007'32                            STA     CG_PNT          ;STORE IN CG_PNT.
7051
7051 C9                                 RET

7052                    ;*****************************************************************
7052                    ;
7052                    ;  INC_CG_PNT  THIS ROUTINE INCREMENTS THE COUNTER GROUP POINTER.
7052                    ;
7052                    ;*****************************************************************

7052 00'07'21           INC_CG_PNT:     LXI     H,CG_PNT        ;POINT TO CG_PNT.
7055 34                                 INR     M               ;INCREMENT CG_PNT.
7056
7056 C9                                 RET

7057                    ;*****************************************************************
7057                    ;
7057                    ;  WRITE_MASTER  THIS ROUTINE WRITE THE DATA, POINTED TO BY THE H,L PAIR,
7057                    ;                TO THE MASTER MODE REGISTER.
7057                    ;
7057                    ;*****************************************************************

7057 17 3E              WRITE_MASTER:   MVI     A,MASTER_MODE   ;POINT TO THE MASTER MODE REG.
7059 1D D3                              OUT     ITCSR
705B
705B 00D9'CD                            CALL    WRITE_ITDB      ;WRITE THE DATA TO THE MASTER REG.
705E
```

```
705E  480 C9                                  RET
705F  481
705F  482
705F  483                          ;*****************************************************************************
705F  484                          ;
705F  485                          ;  READ_MASTER  THIS ROUTINE READS THE MASTER MODE REGISTER AND PUTS THE
705F  486                          ;               DATA IN REC_DAT.  IT THEN REWRITES THE MASTER MODE REG TO
705F  487                          ;               NORMAL OPERATION (WRITES 0A00 INTO MASTER MODE REG).
705F  488                          ;
705F  489                          ;*****************************************************************************
705F  490
705F  491
705F  492 17 3E                    READ_MASTER:    MVI     A,MASTER_MODE   ;POINT TO THE MASTER MODE REG.
7061  493 1D D3                                    OUT     ITCSR
7063  494
7063  495 00E2'CD                                  CALL    READ_ITDB       ;READ THE MASTER MODE REGISTER.
7066  496
7066  497 00'0C'21                                LXI     H,NORMAL_MASTER ;POINT TO 0A00 DATA
7069  498 0057'CD                                  CALL    WRITE_MASTER    ;WRITE NORMAL MASTER REG DATA
706C  499
706C  500 C9                                      RET
706D  501
706D  502
706D  503                          ;*****************************************************************************
706D  504                          ;
706D  505                          ;  WRITE_MODE   THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR,
706D  506                          ;               TO THE MODE REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
706D  507                          ;
706D  508                          ;*****************************************************************************
706D  509
706D  510
706D  511 00 06                    WRITE_MODE:     MVI     B,MODE_REG      ;DATA FOR MODE REGISTER WITH COUNTER
706F  512                                                                  ; GROUP FIELD CLEARED.
706F  513
706F  514 0007'3A                                 LDA     CG_PNT          ;OR IN THE COUNTER GROUP POINTER.
7072  515 B0                                      ORA     B
7073  516 1D D3                                   OUT     ITCSR           ;POINT TO THE CGX MODE REGISTER.
7075  517
7075  518 00D9'CD                                 CALL    WRITE_ITDB      ;WRITE THE DATA TO THE MODE REGISTER.
7078  519
7078  520 C9                                      RET
7079  521
7079  522
7079  523                          ;*****************************************************************************
7079  524                          ;
7079  525                          ;  READ_MODE    THIS ROUTINE READS THE DATA FROM THE MODE REGISTER POINTED TO
7079  526                          ;               BY THE COUNTER GROUP POINTER AND PUTS IT IN REC_DAT.
7079  527                          ;
7079  528                          ;*****************************************************************************
7079  529
7079  530
7079  531 00 06                    READ_MODE:      MVI     B,MODE_REG      ;DATA FOR MODE REGISTER WITH COUNTER
707B  532                                                                  ; GROUP FIELD CLEARED.
707B  533
707B  534 0007'3A                                 LDA     CG_PNT          ;OR IN THE COUNTER GROUP POINTER.
707E  535 B0                                      ORA     B
707F  536 1D D3                                   OUT     ITCSR           ;POINT TO THE CGX MODE REGISTER.
7081  537
7081  538 00E2'CD                                 CALL    READ_ITDB       ;READ THE COUNTER MODE REGISTER.
7084  539
```

```
7084  540 C9                                  RET
7085  541
7085  542
7085  543                          ;*****************************************************************************
7085  544                          ;
7085  545                          ;  WRITE_LOAD  THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR,
7085  546                          ;              TO THE LOAD REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
7085  547                          ;
7085  548                          ;*****************************************************************************
7085  549
7085  550
7085  551 08 06                    WRITE_LOAD:     MVI     B,LOAD_REG      ;DATA FOR LOAD REGISTER WITH COUNTER
7087  552                                                                   ; GROUP FIELD CLEARED.
7087  553
7087  554 0007'3A                                  LDA     CG_PNT          ;OR IN THE COUNTER GROUP POINTER.
708A  555 B0                                      ORA     B
708B  556 1D D3                                   OUT     ITCSR           ;POINT TO THE CGX LOAD REGISTER.
708D  557
708D  558 00D9'CD                                 CALL    WRITE_ITDB      ;WRITE THE DATA TO THE LOAD REGISTER.
7090  559
7090  560 C9                                      RET
7091  561
7091  562                          ;*****************************************************************************
7091  563                          ;
7091  564                          ;  READ_LOAD  THIS ROUTINE READS THE DATA FROM THE LOAD REGISTER POINTED TO
7091  565                          ;             BY THE COUNTER GROUP POINTER AND PUTS IT IN REC_DAT.
7091  566                          ;
7091  567                          ;*****************************************************************************
7091  568
7091  569
7091  570 08 06                    READ_LOAD:      MVI     B,LOAD_REG      ;DATA FOR LOAD REGISTER WITH COUNTER
7093  571                                                                   ; GROUP FIELD CLEARED.
7093  572
7093  573 0007'3A                                  LDA     CG_PNT          ;OR IN THE COUNTER GROUP POINTER.
7096  574 B0                                      ORA     B
7097  575 1D D3                                   OUT     ITCSR           ;POINT TO THE CGX LOAD REGISTER.
7099  576
7099  577 00E2'CD                                 CALL    READ_ITDB       ;READ THE DATA FROM THE LOAD REGISTER.
709C  578
709C  579 C9                                      RET
709D  580
709D  581
709D  582                          ;*****************************************************************************
709D  583                          ;
709D  584                          ;  WRITE_HOLD  THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR,
709D  585                          ;              TO THE HOLD REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
709D  586                          ;
709D  587                          ;*****************************************************************************
709D  588
709D  589
709D  590 10 06                    WRITE_HOLD:     MVI     B,HOLD_REG      ;DATA FOR HOLD REGISTER WITH COUNTER
709F  591                                                                   ; GROUP FIELD CLEARED.
709F  592
709F  593 0007'3A                                  LDA     CG_PNT          ;OR IN THE COUNTER GROUP POINTER.
70A2  594 B0                                      ORA     B
70A3  595 1D D3                                   OUT     ITCSR           ;POINT TO THE CGX HOLD REGISTER.
70A5  596
70A5  597 00D9'CD                                 CALL    WRITE_ITDB      ;WRITE THE DATA TO THE HOLD REGISTER.
70A8  598
70A8  599 C9                                      RET
```

```
70A9                                  ;*****************************************************************************
70A9                                  ;
70A9                                  ;  READ_HOLD  THIS ROUTINE READS THE DATA FROM THE HOLD REGISTER POINTED TO
70A9                                  ;             BY THE COUNTER GROUP POINTER AND PUTS IT IN REC_DAT.
70A9                                  ;
70A9                                  ;*****************************************************************************
70A9
70A9 10 06                            READ_HOLD:    MVI     B,HOLD_REG      ;DATA FOR HOLD REGISTER WITH COUNTER
70AB                                                                        ; GROUP FIELD CLEARED.
70AB
70AB 0007'3A                                        LDA     CG_PNT          ;OR IN THE COUNTER GROUP POINTER.
70AE B0                                             ORA     B
70AF 1D D3                                          OUT     ITCSR           ;POINT TO THE CGX HOLD REGISTER.
70B1
70B1 00E2'CD                                        CALL    READ_ITDB       ;READ THE DATA FROM THE HOLD REGISTER.
70B4
70B4 C9                                             RET
70B5
70B5
70B5                                  ;*****************************************************************************
70B5                                  ;
70B5                                  ;  WRITE_ALARM  THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR, TO
70B5                                  ;               THE ALARM REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
70B5                                  ;
70B5                                  ;*****************************************************************************
70B5
70B5 07 06                            WRITE_ALARM:  MVI     B,ALARM_REG1    ;DATA FOR ALARM REGISTER 1.
70B7
70B7 0007'3A                                        LDA     CG_PNT          ;ARE WE POINTING TO COUNTER GROUP 1.
70BA 3D                                             DCR     A
70BB 00C0'CA                                        JZ      1$              ;SKIP NEXT INSTRUCTION IF SO.
70BE
70BE 0F 06                                          MVI     B,ALARM_REG2    ;NO, LOAD DATA FOR ALARM REGISTER 2.
70C0 78                               1$:           MOV     A,B             ;MOVE ALARM DATA INTO A.
70C1
70C1 1D D3                                          OUT     ITCSR           ;POINT TO THE CGX ALARM REGISTER.
70C3
70C3 00D9'CD                                        CALL    WRITE_ITDB      ;WRITE THE DATA TO THE ALARM REG.
70C6
70C6 C9                                             RET
70C7
70C7
70C7                                  ;*****************************************************************************
70C7                                  ;
70C7                                  ;  READ_ALARM  THIS ROUTINE WRITES THE DATA, POINTED TO BY THE H,L PAIR, TO
70C7                                  ;              THE ALARM REGISTER POINTED TO BY THE COUNTER GROUP POINTER.
70C7                                  ;
70C7                                  ;*****************************************************************************
70C7
70C7 07 06                            READ_ALARM:   MVI     B,ALARM_REG1    ;DATA FOR ALARM REGISTER 1.
70C9
70C9 0007'3A                                        LDA     CG_PNT          ;ARE WE POINTING TO COUNTER GROUP 1.
70CC 3D                                             DCR     A
70CD 00D2'CA                                        JZ      1$              ;SKIP NEXT INSTRUCTION IF SO.
70D0
```

```
70D0  660 0F 06                 MVI     B,ALARM_REG2    ;NO, LOAD DATA FOR ALARM REGISTER 2.
70D2  661 78            1$:     MOV     A,B             ;MOVE ALARM DATA INTO A.
70D3  662
70D3  663 1D D3                 OUT     ITCSR           ;POINT TO THE CGX ALARM REGISTER.
70D5  664
70D5  665 00E2'CD               CALL    READ_ITDB       ;READ THE ALARM REGISTER.
70D8  666
70D8  667 C9                    RET
70D9  668
70D9  669
70D9  670               ;*****************************************************************
70D9  671               ;
70D9  672               ;  WRITE_ITDB  THIS ROUTINE WRITES TWO BYTES OF DATA POINTED TO BY THE H,L PAIR
70D9  673               ;              TO THE INTERVAL TIMER DATA BUS. THE DATA POINTER REGISTER SHOULD
70D9  674               ;              BE LOADED PRIOR TO USING THIS ROUTINE.
70D9  675               ;
70D9  676               ;*****************************************************************
70D9  677
70D9  678
70D9  679 23            WRITE_ITDB:     INX     H               ;WRITE THE LOW BYTE TO THE
70DA  680 7E                            MOV     A,M             ; INTERVAL TIMER.
70DB  681 1C D3                         OUT     ITDB
70DD  682
70DD  683 2B                            DCX     H               ;WRITE THE HIGH BYTE.
70DE  684 7E                            MOV     A,M
70DF  685 1C D3                         OUT     ITDB
70E1  686
70E1  687 C9                            RET
70E2  688
70E2  689
70E2  690               ;*****************************************************************
70E2  691               ;
70E2  692               ;  READ_ITDB   THIS ROUTINE READS TWO BYTES OF DATA FROM THE INTERVAL TIMER DATA
70E2  693               ;              BUS TO REC_DAT. THE DATA POINTER REGISTER SHOULD BE LOADED PRIOR
70E2  694               ;              TO USING THIS ROUTINE.
70E2  695               ;
70E2  696               ;*****************************************************************
70E2  697
70E2  698
70E2  699 1C DB         READ_ITDB:      IN      ITDB            ;READ THE LOW BYTE AND PUT IT IN
70E4  700 002C'32                       STA     REC_DAT+1       ; REC_DAT+1.
70E7  701
70E7  702 1C DB                         IN      ITDB            ;READ THE HIGH BYTE AND PUT IT IN
70E9  703 002B'32                       STA     REC_DAT         ; REC_DAT.
70EC  704
70EC  705 C9                            RET
70ED  706
70ED  707
70ED  708               ;*****************************************************************
70ED  709               ;
70ED  710               ;  MM_BITS     THIS ROUTINE MOVES TEMP_DAT TO EXP_DAT AND SETS BIT 14 (DISABLE
70ED  711               ;              DATA POINTER INCREMENT) AND CLEARS BIT 13 (8-BIT DATA BUS) OF
70ED  712               ;              EXP_DAT. TEMP_DAT IS UNCHANGED.
70ED  713               ;
70ED  714               ;*****************************************************************
70ED  715
70ED  716
70ED  717 00'25'21      MM_BITS:        LXI     H,TEMP_DAT      ;POINT TO TEMP_DAT.
70F0  718 7E                            MOV     A,M             ;MOVE HIGH BYTE INTO A.
70F1  719 40 F6                         ORI     <^X40>          ;SET BIT 14.
```

```
70F3  720 DF E6                    ANI     <^XDF>          ;CLEAR BIT 13.
70F5  721 00'28'21                 LXI     H,EXP_DAT
70F8  722 77                       MOV     M,A             ;MOVE HIGH BYTE BACK TO EXP_DAT.
70F9  723
70F9  724 0026'3A                  LDA     TEMP_DAT+1      ;MOVE LOW BYTE OF TEMP_DAT
70FC  725 0029'32                  STA     EXP_DAT+1       ; INTO LOW BYTE OF EXP_DAT.
70FF  726
70FF  727 C9                       RET
7100  728
7100  729
7100  730                 ;*********************************************************************
7100  731                 ;
7100  732                 ;  INC_CYCLE_CNT THIS ROUTINE INCREMENTS THE VALUE IN CYCLE_CNT.
7100  733                 ;
7100  734                 ;*********************************************************************
7100  735
7100  736
7100  737 00'0B'21       INC_CYCLE_CNT:  LXI     H,CYCLE_CNT     ;POINT TO CYCLE_CNT.
7103  738 34                       INR     M               ;INCREMENT IT.
7104  739
7104  740 C9                       RET
7105  741
7105  742
7105  743                 ;*********************************************************************
7105  744                 ;
7105  745                 ;  DISARM_CNTR  THIS ROUTINE DISARMS THE COUNTER SPECIFIED BY CG_PNT.
7105  746                 ;
7105  747                 ;*********************************************************************
7105  748
7105  749
7105  750 0007'3A        DISARM_CNTR:    LDA     CG_PNT          ;LOAD THE COUNTER GROUP POINTER IN A.
7108  751 0008'32                  STA     SHIFT_CNT       ;STORE A IN SHIFT_CNT.
710B  752
710B  753 AF                       XRA     A               ;CLEAR A.
710C  754 3F                       CMC                     ;SET THE CARRY.
710D  755
710D  756 17                1$:    RAL                     ;SHIFT A THROUGH CARRY.
710E  757
710E  758 00'08'21                 LXI     H,SHIFT_CNT     ;POINT TO SHIFT_CNT.
7111  759 35                       DCR     M               ;DECREMENT SHIFT_CNT.
7112  760 010D'C2                  JNZ     1$              ;IF ZERO, STOP SHIFTING.
7115  761
7115  762 C0 F6                    ORI     DISARM_CNTR_DAT ;OR IN DISARM COUNTER DATA.
7117  763
7117  764 1D D3                    OUT     ITCSR           ;DISARM COUNTER.
7119  765
7119  766 C9                       RET
711A  767
711A  768
711A  769
711A  770                 ;*********************************************************************
711A  771                 ;
711A  772                 ;  LOAD_CNTR  THIS ROUTINE LOADS THE COUNTER OF THE COUNTER GROUP SPECIFIED BY
711A  773                 ;             CG_PNT FROM THE SOURCE SPECIFIED BY THE COUNTER MODE REGISTER.
711A  774                 ;
711A  775                 ;*********************************************************************
711A  776
711A  777
711A  778 0007'3A        LOAD_CNTR:      LDA     CG_PNT
711D  779 0008'32                  STA     SHIFT_CNT       ;STORE A IN SHIFT_CNT.
```

```
7120                                        XRA     A               ;CLEAR A.
7121  AF                                    CMC                     ;SET THE CARRY.
7122  3F
7122                                1$:     RAL                     ;SHIFT A THROUGH CARRY.
7123  17
7123                                        LXI     H,SHIFT_CNT     ;POINT TO SHIFT_CNT.
7126  00'08'21                              DCR     M               ;DECREMENT SHIFT_CNT.
7127  35                                    JNZ     1$              ;IF ZERO, STOP SHIFTING.
712A  0122'C2
712A                                        ORI     LOAD_CNTR_DAT   ;OR IN LOAD COUNTER DATA.
712C  40 F6
712C                                        OUT     ITCSR           ;LOAD COUNTER FROM REGISTER SPECIFIED
712E  1D D3                                                         ; BY COUNTER MODE REGISTER.
712E
712E                                        RET
712E  C9
712F
712F                                ;*******************************************************************
712F                                ;
712F                                ;  SAVE_CNTR  THIS ROUTINE SAVES THE COUNTER SPECIFIED BY CG_PNT IN THE
712F                                ;            CORRESPONDING HOLD REGISTER.
712F                                ;
712F                                ;*******************************************************************
712F
712F                                SAVE_CNTR:  LDA     CG_PNT          ;LOAD THE COUNTER GROUP POINTER IN A.
712F  0007'3A                               STA     SHIFT_CNT       ;STORE A IN SHIFT_CNT.
7132  0008'32
7135                                        XRA     A               ;CLEAR A.
7135  AF                                    CMC                     ;SET THE CARRY.
7136  3F
7137                                1$:     RAL                     ;SHIFT A THROUGH CARRY.
7137  17
7138                                        LXI     H,SHIFT_CNT     ;POINT TO SHIFT_CNT.
7138  00'08'21                              DCR     M               ;DECREMENT SHIFT_CNT.
713B  35                                    JNZ     1$              ;IF ZERO, STOP SHIFTING.
713C  0137'C2
713F                                        ORI     SAVE_CNTR_DAT   ;OR IN SAVE COUNTER DATA.
713F  A0 F6
7141                                        OUT     ITCSR           ;SAVE COUNTER IN CORRESPONDING
7141  1D D3                                                         ; SAVE REGISTER.
7143
7143                                        RET
7143  C9
7144
7144                                ;*******************************************************************
7144                                ;
7144                                ;  SET_OUTPUT  THIS ROUTINE SET THE OUTPUT BIT OF THE COUNTER GROUP SPECIFIED
7144                                ;             BY CG_PNT.
7144                                ;
7144                                ;*******************************************************************
7144
7144                                SET_OUTPUT: LXI     H,CG_PNT        ;POINT TO THE COUNTER GROUP.
7144  00'07'21                              MVI     A,SET_OUTPUT_DAT ;OR IN THE SET OUTPUT DATA.
7147  E8 3E                                 ORA     M
7149  B6
714A                                        OUT     ITCSR           ;SET THE OUTPUT BIT.
714A  1D D3
714C
```

```
714C  840 C9                                  RET
714D  841
714D  842
714D  843                                ;*****************************************************************************
714D  844                                ;
714D  845                                ;  CLR_OUTPUT  THIS ROUTINE CLEARS THE OUTPUT BIT OF THE COUNTER GROUP
714D  846                                ;              SPECIFIED BY CG_PNT.
714D  847                                ;
714D  848                                ;*****************************************************************************
714D  849
714D  850
714D  851 00'07'21                       CLR_OUTPUT:    LXI     H,CG_PNT           ;POINT TO THE COUNTER GROUP.
7150  852 E0 3E                                         MVI     A,CLR_OUTPUT_DAT   ;OR IN THE CLEAR OUTPUT DATA.
7152  853 B6                                            ORA     M
7153  854
7153  855 1D D3                                         OUT     ITCSR              ;SET THE OUTPUT BIT.
7155  856
7155  857 C9                                            RET
7156  858
7156  859
7156  860                                ;*****************************************************************************
7156  861                                ;
7156  862                                ;  INC_CNTR  THIS ROUTINE INCREMENTS THE COUNTER SPECIFIED BY CG_PNT.
7156  863                                ;
7156  864                                ;*****************************************************************************
7156  865
7156  866
7156  867 00'07'21                       INC_CNTR:      LXI     H,CG_PNT           ;POINT TO THE COUNTER GROUP.
7159  868 F0 3E                                         MVI     A,INC_CNTR_DAT     ;OR IN THE INCREMENT COUNTER
715B  869 B6                                            ORA     M                  ; DATA.
715C  870
715C  871 1D D3                                         OUT     ITCSR              ;INCREMENT THE COUNTER.
715E  872
715E  873 C9                                            RET
715F  874
715F  875
715F  876                                ;*****************************************************************************
715F  877                                ;
715F  878                                ;  LOAD_MOD_DAT  THIS ROUTINE LOADS THE MODULE DATA POINTED TO BY THE H,L
715F  879                                ;                PAIR INTO CON_ADD_DAT.
715F  880                                ;
715F  881                                ;*****************************************************************************
715F  882
715F  883
715F  884 00'00'11                       LOAD_MOD_DAT:  LXI     D,CON_MOD_DAT      ;LOAD THE MODULE DATA POINTED TO BY
7162  885 05 0E                                         MVI     C,5                ; THE H,L PAIR INTO CON_MOD_DAT.
7164  886 0000'CD                                       CALL    MOVER
7167  887
7167  888 C9                                            RET
7168  889
7168  890
7168  891                                ;*****************************************************************************
7168  892                                ;
7168  893                                ;  TEST1E_ERROR  THIS ROUTINE IS USED BY TEST 1E TO DEFINE THE ERROR VARIABLES
7168  894                                ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
7168  895                                ;
7168  896                                ;*****************************************************************************
7168  897
7168  898
7168  899 0028'2A                        TEST1E_ERROR:  LHLD    EXP_DAT            ;MOVE EXPECTED DATA TO CON_EXP_DAT.
```

```
716B  900 0002'22                    SHLD    CON_EXP_DAT+2
716E  901
716E  902 002B'2A                    LHLD    REC_DAT          ;MOVE RECEIVED DATA TO CON_REC_DAT.
7171  903 0002'22                    SHLD    CON_REC_DAT+2
7174  904
7174  905 FF FF 21                   LXI     H,<^XFFFF>       ;LAST TWO BYTES ARE OF INTEREST.
7177  906 0000'22                    SHLD    CON_MSK_DAT
717A  907 02 0E 00'02'21             ZERO    CON_MSK_DAT+2,2
          0D 23 77 AF
          0180'C2
7186  908
7186  909 03 0E 00'00'21             ZERO    CON_ADD_DAT,3    ;STORE COUNTER GORUP POINTER
          0D 23 77 AF
          018C'C2
7192  910 0007'3A                    LDA     CG_PNT           ; IN ADDITIONAL DATA.
7195  911 0003'32                    STA     CON_ADD_DAT+3
7198  912
7198  913 00'00'21                   LXI     H,MWCS_A         ;WCS MODULE BEING TESTED.
719B  914 015F'CD                    CALL    LOAD_MOD_DAT
719E  915
719E  916 0000'CD                    CALL    CON_ERROR        ; PRINT ERROR MESSAGE.
71A1  917
71A1  918 C9                         RET
71A2  919
71A2  920
71A2  921                    ;*****************************************************************
71A2  922                    ;
71A2  923                    ;  TEST1F_ERROR  THIS ROUTINE IS USED BY TEST 1F TO DEFINE THE ERROR VARIABLES
71A2  924                    ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
71A2  925                    ;
71A2  926                    ;*****************************************************************
71A2  927
71A2  928
71A2  929 01 3E              TEST1F_ERROR:   MVI     A,1              ;ERROR NUMBER 1.
71A4  930 0000'32                            STA     CON_ERR_NUM
71A7  931
71A7  932 0028'2A                            LHLD    EXP_DAT          ;MOVE EXPECTED DATA TO CON_EXP_DAT.
71AA  933 0002'22                            SHLD    CON_EXP_DAT+2
71AD  934
71AD  935 002B'2A                            LHLD    REC_DAT          ;MOVE RECEIVED DATA TO CON_REC_DAT.
71B0  936 0002'22                            SHLD    CON_REC_DAT+2
71B3  937
71B3  938 FF FF 21                           LXI     H,<^XFFFF>       ;BITS 0-13 AND 15 ARE OF INTEREST.
71B6  939 0000'22                            SHLD    CON_MSK_DAT
71B9  940 00 60 21                           LXI     H,<^X0060>
71BC  941 0002'22                            SHLD    CON_MSK_DAT+2
71BF  942
71BF  943 0000'32 3C AF                      SET     NA_OTHER         ;NO ADDITIONAL DATA.
71C4  944
71C4  945 00'00'21                           LXI     H,MWCS_A         ;WCS MODULE BEING TESTED.
71C7  946 015F'CD                            CALL    LOAD_MOD_DAT
71CA  947
71CA  948 0000'CD                            CALL    CON_ERROR        ; PRINT ERROR MESSAGE.
71CD  949
71CD  950 C9                                 RET
71CE  951
71CE  952
71CE  953                    ;*****************************************************************
71CE  954                    ;
71CE  955                    ;  TEST21_ERROR  ROUTINE USED BY TESTS 21,23 AND 25 TO DEFINE THE ERROR
```

```
71CE  956
71CE  957
71CE  958
71CE  959
71CE  960
71CE  961 01 3E
71D0  962 0000'32
71D3  963
71D3  964 0028'2A
71D6  965 0002'22
71D9  966
71D9  967 002B'2A
71DC  968 0002'22
71DF  969
71DF  970 FF FF 21
71E2  971 0000'22
71E5  972 02 0E 00'02'21
          0D 23 77 AF
          01EB'C2
71F1  973
71F1  974 03 0E 00'00'21
          0D 23 77 AF
          01F7'C2
71FD  975 0007'3A
7200  976 0003'32
7203  977
7203  978 00'00'21
7206  979 015F'CD
7209  980
7209  981 0000'CD
720C  982
720C  983 C9
720D  984
720D  985
720D  986
720D  987
720D  988
720D  989
720D  990
720D  991
720D  992
720D  993
720D  994 0028'2A
7210  995 0002'22
7213  996
7213  997 002B'2A
7216  998 0002'22
7219  999
7219 1000 FF FF 21
721C 1001 0000'22
721F 1002 02 0E 00'02'21
          0D 23 77 AF
          0225'C2
722B 1003
722B 1004 03 0E 00'00'21
          0D 23 77 AF
          0231'C2
7237 1005 000B'3A
723A 1006 0003'32
723D 1007
```

```
;                VARIABLES IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
;
;*****************************************************************************

TEST21_ERROR:   MVI     A,1             ;ERROR NUMBER 1.
                STA     CON_ERR_NUM

                LHLD    EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT.
                SHLD    CON_EXP_DAT+2

                LHLD    REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT.
                SHLD    CON_REC_DAT+2

                LXI     H,<^XFFFF>      ;LAST TWO BYTES ARE OF INTEREST.
                SHLD    CON_MSK_DAT
                ZERO    CON_MSK_DAT+2,2


                ZERO    CON_ADD_DAT,3   ;ADDITIONAL DATA CONTAINS COUNTER

                LDA     CG_PNT          ; GROUP POINTER.
                STA     CON_ADD_DAT+3

                LXI     H,MWCS_A        ;WCS MODULE BEING TESTED.
                CALL    LOAD_MOD_DAT

                CALL    CON_ERROR       ; PRINT ERROR MESSAGE.

                RET


;*****************************************************************************
;
;  TEST22_ERROR  THIS ROUTINE IS USED BY TEST 22 TO DEFINE THE ERROR VARIABLES
;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
;
;*****************************************************************************

TEST22_ERROR:   LHLD    EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT.
                SHLD    CON_EXP_DAT+2

                LHLD    REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT.
                SHLD    CON_REC_DAT+2

                LXI     H,<^XFFFF>      ;LAST TWO BYTES ARE OF INTEREST.
                SHLD    CON_MSK_DAT
                ZERO    CON_MSK_DAT+2,2


                ZERO    CON_ADD_DAT,3   ;STORE THE CYCLE COUNTER IN

                LDA     CYCLE_CNT       ; CON_ADD_DAT.
                STA     CON_ADD_DAT+3
```

```
723D 1008 00'00'21                  LXI     H,MWCS_A        ;WCS MODULE BEING TESTED.
7240 1009 015F'CD                   CALL    LOAD_MOD_DAT
7243 1010
7243 1011 0000'CD                   CALL    CON_ERROR       ; PRINT ERROR MESSAGE.
7246 1012
7246 1013 C9                        RET
7247 1014
7247 1015
7247 1016                   ;*******************************************************************
7247 1017                   ;
7247 1018                   ;  TEST24_ERROR  THIS ROUTINE IS USED BY TEST 24 TO DEFINE THE ERROR VARIABLES
7247 1019                   ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
7247 1020                   ;
7247 1021                   ;*******************************************************************
7247 1022
7247 1023
7247 1024 0028'3A           TEST24_ERROR:   LDA     EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT.
724A 1025 0003'32                           STA     CON_EXP_DAT+3
724D 1026
724D 1027 002B'3A                           LDA     REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT.
7250 1028 0003'32                           STA     CON_REC_DAT+3
7253 1029
7253 1030 0000'32 3C AF                     SET     NA_OTHER        ;NO ADDITIONAL DATA.
7258 1031
7258 1032 FF FF 21                          LXI     H,<^XFFFF>      ;LAST BYTE IS OF INTEREST.
725B 1033 0000'22                           SHLD    CON_MSK_DAT
725E 1034 00 FF 21                          LXI     H,<^X00FF>
7261 1035 0002'22                           SHLD    CON_MSK_DAT+2
7264 1036
7264 1037 00'00'21                          LXI     H,MWCS_A        ;WCS MODULE BEING TESTED.
7267 1038 015F'CD                           CALL    LOAD_MOD_DAT
726A 1039
726A 1040 0000'CD                           CALL    CON_ERROR       ; PRINT ERROR MESSAGE.
726D 1041
726D 1042 C9                                RET
726E 1043
726E 1044
726E 1045                   ;*******************************************************************
726E 1046                   ;
726E 1047                   ;  TEST26_ERROR  THIS ROUTINE IS USED BY TEST 26 TO DEFINE THE ERROR VARIABLES
726E 1048                   ;                IN PREPARATION FOR USING THE CON_ERROR ROUTINE.
726E 1049                   ;
726E 1050                   ;*******************************************************************
726E 1051
726E 1052
726E 1053 01 3E             TEST26_ERROR:   MVI     A,1             ;ERROR NUMBER 1.
7270 1054 0000'32                           STA     CON_ERR_NUM
7273 1055
7273 1056 0028'3A                           LDA     EXP_DAT         ;MOVE EXPECTED DATA TO CON_EXP_DAT.
7276 1057 0003'32                           STA     CON_EXP_DAT+3
7279 1058
7279 1059 002B'3A                           LDA     REC_DAT         ;MOVE RECEIVED DATA TO CON_REC_DAT.
727C 1060 0003'32                           STA     CON_REC_DAT+3
727F 1061
727F 1062 02 0E 00'00'21                    ZERO    CON_ADD_DAT,2   ;ADDITONAL DATA CONTAINS DATA SENT
          0D 23 77 AF
          0285'C2
728B 1063 0025'2A                           LHLD    TEMP_DAT        ; TO THE ALARM REGISTER.
728E 1064 0002'22                           SHLD    CON_ADD_DAT+2
7291 1065
```

```
7291 1066 FF FF 21                          LXI     H,<^XFFFF>        ;DATA OF INTEREST IS BITS 1 AND 2.
7294 1067 0000'22                           SHLD    CON_MSK_DAT
7297 1068 F9 FF 21                          LXI     H,<^XF9FF>
729A 1069 0002'22                           SHLD    CON_MSK_DAT+2
729D 1070
729D 1071 00'00'21                          LXI     H,MWCS_A          ;WCS MODULE BEING TESTED.
72A0 1072 015F'CD                           CALL    LOAD_MOD_DAT
72A3 1073
72A3 1074 0000'CD                           CALL    CON_ERROR         ; PRINT ERROR MESSAGE.
72A6 1075
72A6 1076 C9                                RET
72A7 1077
72A7 1078                         ;*****************************************************************************
72A7 1079                         ;
72A7 1080                         ;  TEST20_ONES_ZEROS    THIS ROUTINE IS USED BY TEST 20 TO CHECK THE MODE,
72A7 1081                         ;                       LOAD, AND HOLD REGISTERS WITH DATA PATTERNS OF ALL
72A7 1082                         ;                       ZEROS OR ALL ONES.  EXP_DAT POINTS TO ZEROS OR ONES
72A7 1083                         ;                       DATA.
72A7 1084                         ;
72A7 1085                         ;*****************************************************************************
72A7 1086
72A7 1087                         TEST20_ONES_ZEROS:
72A7 1088 00'28'21                          LXI     H,EXP_DAT         ;WRITE EXP_DAT TO THE COUNTER MODE
72AA 1089 006D'CD                           CALL    WRITE_MODE        ; REGISTER.
72AD 1090
72AD 1091 0079'CD                           CALL    READ_MODE         ;READ THE COUNTER MODE REGISTER.
72B0 1092
72B0 1093 00'28'21          ******************  IFNEQ   EXP_DAT,REC_DAT,2         ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11    *
          02C6'CA 0000'CD   *
72BE 1094                   *                                                     ; RECEIVED DATA.
72BE 1095                   *
72BE 1096 01 3E             *                       MVI     A,1               ;ERROR NUMBER 1.
72C0 1097 0000'32           *                       STA     CON_ERR_NUM
72C3 1098                   *
72C3 1099 0168'CD           *                       CALL    TEST1E_ERROR      ;PRINT AN ERROR MESSAGE.
72C6 1100                   *
72C6 1101                   ******************      ENDIF
72C6 1102
72C6 1103 0000'3A                                   LDA     ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
72C9 1104 0F                                        RRC
72CA 1105 02A7'DA                                   JC      TEST20_ONES_ZEROS ;...JUMP TO BEGINNING OF ERROR LOOP.
72CD 1106
72CD 1107 00'28'21                  1$:             LXI     H,EXP_DAT         ;WRITE EXP_DAT TO THE COUNTER LOAD
72D0 1108 0085'CD                                   CALL    WRITE_LOAD        ; REGISTER.
72D3 1109
72D3 1110 0091'CD                                   CALL    READ_LOAD         ;READ THE COUNTER LOAD REGISTER.
72D6 1111
72D6 1112 00'28'21          ******************      IFNEQ   EXP_DAT,REC_DAT,2         ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11    *
          02EC'CA 0000'CD   *
72E4 1113                   *                                                     ; RECEIVED DATA.
72E4 1114                   *
72E4 1115 02 3E             *                       MVI     A,2               ;ERROR NUMBER 2.
72E6 1116 0000'32           *                       STA     CON_ERR_NUM
72E9 1117                   *
72E9 1118 0168'CD           *                       CALL    TEST1E_ERROR      ;PRINT AN ERROR MESSAGE.
72EC 1119                   *
72EC 1120                   ******************      ENDIF
72EC 1121
```

```
72EC 1122 0000'3A                                  LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
72EF 1123 0F                                       RRC
72F0 1124 02CD'DA                                  JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
72F3 1125
72F3 1126
72F3 1127 00'28'21                         2$:     LXI     H,EXP_DAT       ;WRITE EXP_DAT TO THE COUNTER HOLD
72F6 1128 009D'CD                                  CALL    WRITE_HOLD      ; REGISTER.
72F9 1129
72F9 1130 00A9'CD                                  CALL    READ_HOLD       ;READ THE COUNTER HOLD REGISTER.
72FC 1131
72FC 1132 00'28'21          *****************      IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11    *
          0312'CA 0000'CD   *
730A 1133                   *                                                  ; RECEIVED DATA.
730A 1134                   *
730A 1135 03 3E             *                      MVI     A,3             ;ERROR NUMBER 3.
730C 1136 0000'32           *                      STA     CON_ERR_NUM
730F 1137                   *
730F 1138 0168'CD           *                      CALL    TEST1E_ERROR    ;PRINT AN ERROR MESSAGE.
7312 1139                   *
7312 1140                   *****************      ENDIF
7312 1141
7312 1142 0000'3A                                  LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7315 1143 0F                                       RRC
7316 1144 02F3'DA                                  JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7319 1145
7319 1146 C9                                       RET                     ;DONE WITH ALL ONES OR ALL ZEROS
731A 1147
731A 1148
731A 1149                          ;*****************************************************************************
731A 1150                          ;
731A 1151                          ;  INTERVAL TIMER TESTS - THE NEXT 14 TESTS DEAL SPECIFICALLY WITH THE
731A 1152                          ;                         INTERVAL TIMER CHIP ON THE WCS BOARD.
731A 1153                          ;
731A 1154                          ;*****************************************************************************
731A 1155
731A 1156
731A 1157                          ;*****************************************************************************
731A 1158                          ;
731A 1159                          ;
731A 1160                          ;       TEST1E                  MASTER RESET TEST
731A 1161                          ;       ------                  -----------------
731A 1162                          ;
731A 1163                          ;
731A 1164                          ;               THIS TEST PERFORMS AN INTERVAL TIMER MASTER RESET BY WRITING
731A 1165                          ;       A HEX FF TO THE COMMAND REGISTER. EACH OF THE REGISTERS IS THEN READ
731A 1166                          ;       TO SEE THAT IT HAS BEEN PROPERLY INITAILIZED. THE FOLLOWING VALUES
731A 1167                          ;       SHOULD BE FOUND IN EACH REGISTER.
731A 1168                          ;
731A 1169                          ;
731A 1170                          ;               REGISTER                        CONTENTS
731A 1171                          ;               --------                        --------
731A 1172                          ;
731A 1173                          ;       (5) 16-BIT COUNTER MODE REGISTERS        HEX 0B00
731A 1174                          ;       (5) 16 BIT COUNTER LOAD REGISTERS        HEX 0000
731A 1175                          ;       (5) 16 BIT COUNTER HOLD REGISTERS        HEX 0000
731A 1176                          ;       (1) 16-BIT MASTER MODE REGISTER          HEX 0000
731A 1177                          ;
731A 1178                          ;
731A 1179                          ;       TEST STEPS:
```

```
731A 1180
731A 1181                    ;       1) PERFORM A MASTER RESET.
731A 1182                    ;       2) READ EACH REGISTER AND COMPARE EXPECTED AND RECEIVED DATA.
731A 1183                    ;       3) PRINT ERRORS IF ANY.
731A 1184
731A 1185
731A 1186                    ;  ASSUMPTIONS:  BECAUSE THIS IS THE FIRST TEST OF THE INTERVAL TIMER
731A 1187                    ;                CHIP, SEVERAL ASSUMPTIONS MUST BE MADE. IN READING
731A 1188                    ;                THE CONTENTS OF EACH REGISTER, IT MUST BE ASSUMED
731A 1189                    ;                THAT THE DATA POINTER IS WORKING CORRECTLY SO THAT
731A 1190                    ;                THE REGISTER WHICH IS THOUGHT TO BE READ IS THE ONE
731A 1191                    ;                ACTUALLY BEING READ. ALSO, BECAUSE THE CHIP IS BEING
731A 1192                    ;                USED IN 8-BIT MODE, THE BYTE POINTER IS ASSUMED TO BE
731A 1193                    ;                WORKING CORRECTLY, SO THAT THE LOW AND HIGH BYTES OF
731A 1194                    ;                EACH REGISTER ARE BEING READ CORRECTLY.
731A 1195
731A 1196
731A 1197                    ;  ERROR1: MASTER RESET ERROR - MASTER MODE REGISTER.
731A 1198                    ;  ERROR2: MASTER RESET ERROR - MODE REGISTER.
731A 1199                    ;  ERROR3: MASTER RESET ERROR - LOAD REGISTER.
731A 1200                    ;  ERROR4: MASTER RESET ERROR - HOLD REGISTER.
731A 1201
731A 1202                    ;  NOTE: ADDITIONAL DATA WILL GIVE COUNTER GROUP FOR ERRORS 2,3 AND 4.
731A 1203
731A 1204
731A 1205                    ;  DEBUG:  1) CHECK '~WCSA RESET L'' FOR HIGH VALUE DURING MASTER RESET
731A 1206                    ;             IN MASTER RESET ROUTINE.
731A 1207                    ;          2) CHECK '~WCSA SEL TIMER H'' FOR HIGH VALUE DURING RESET. IF
731A 1208                    ;             LOW, CHECK PAL 16L8 (E21).
731A 1209                    ;          3) CHECK CHIP SELECT PIN FOR LOW VALUE DURING RESET. IF
731A 1210                    ;             HIGH CHECK CHIP E17.
731A 1211                    ;          4) CHECK '~WCSA A08 H'' FOR HIGH VALUE DURING RESET.
731A 1212                    ;          5) CHECK '~WCSA WR L'' FOR LOW VALUE DURING RESET
731A 1213                    ;          6) CHECK '~WCSA RD L'' FOR HIGH VALUE DURING RESET.
731A 1214                    ;          7) CHECK DB0-DB7 PINS FOR ALL HIGH VALUES MASTER RESET.
731A 1215                    ;          8) CHECK X2 PIN FOR 1KHZ SIGNAL.
731A 1216                    ;          9) CHECK DB0-DB2 AND DB4 PINS FOR HIGH VALUES DURING WRITE
731A 1217                    ;             TO COMMAND REGISTER IN READ_MASTER ROUTINE.
731A 1218                    ;         10) CHECK '~WCSA A08 H'' FOR LOW VALUE DURING DATA BUS READ
731A 1219                    ;             OF READ_ITDB ROUTINE.
731A 1220                    ;         11) CHECK '~WCSA WR L'' FOR HIGH VALUE AND '~WCSA RD L'' FOR
731A 1221                    ;             LOW VALUE DURING THE SAME READ.
731A 1222                    ;         12) CHECK DB0-DB7 PINS FOR LOW VALUES DURING THIS READ.
731A 1223                    ;             IF INCORRECT, OR IF ERRORS OCCUR READING THE CONTENTS
731A 1224                    ;             OF OTHER REGISTERS, SUSPECT INTERVAL TIMER CHIP.
731A 1225
731A 1226
731A 1227                    ;*****************************************************************************
731A 1228
731A 1229
731A 1230
731A 1231 0000'CD 1E 3E      TEST1E:         HEAD    <^X1E>
          0F 0000'3A
          C9 0327'D2
          0F 0000'3A
          3D D3 0426'DA
7330 1232
7330 1233 01 3E                              MVI     A,1             ;FIRST PART OF TEST 1E.
7332 1234 0000'32                            STA     CON_ERR_NUM
7335 1235
```

```
7335 1236 000E'2A
7338 1237 0028'22
733B 1238
733B 1239 003D'CD                                  1$:
733E 1240
733E 1241 005F'CD
7341 1242
7341 1243 00'28'21              ******************
          02 0E 00'2B'11        *
          0357'CA 0000'CD       *
734F 1244                       *
734F 1245                       *
734F 1246 0000'32 3C AF         *
7354 1247                       *
7354 1248 0168'CD               *
7357 1249                       *
7357 1250                       ******************
7357 1251
7357 1252 0000'3A
735A 1253 0F
735B 1254 033B'DA
735E 1255
735E 1256
735E 1257 02 3E
7360 1258 0000'32
7363 1259
7363 1260 0014'2A
7366 1261 0028'22
7369 1262
7369 1263 004D'CD
736C 1264
736C 1265 C5 46 00'00'21        ******************
          77 05 3E              *
7374 1266                       *
7374 1267 0052'CD               *
7377 1268                       *
7377 1269 0042'CD               *                  2$:
737A 1270                       *
737A 1271                       *
737A 1272 0079'CD               *
737D 1273                       *
737D 1274 00'28'21              * ****************
          02 0E 00'2B'11        * *
          038E'CA 0000'CD       * *
738B 1275                       * *
738B 1276                       * *
738B 1277 0168'CD               * *
738E 1278                       * *
738E 1279                       * ****************
738E 1280                       *
738E 1281 0000'3A               *
7391 1282 0F                    *
7392 1283 0377'DA               *
7395 1284                       *
7395 1285 35 00'00'21           ******************
          70 C1 0374'C2
739E 1286
739E 1287 002D'CD
73A1 1288
73A1 1289 03 3E
```



```
LHLD    ZERO_DAT          ;LOAD THE MASTER MODE EXPECTED DATA
SHLD    EXP_DAT           ; (ALL ZEROS) INTO EXP_DAT.

CALL    MASTER_RESET      ;PERFORM A MASTER RESET.

CALL    READ_MASTER       ;READ THE MASTER MODE REGISTER.

IFNEQ   EXP_DAT,REC_DAT,2         ;COMPARE THE EXPECTED AND

                                  ; RECEIVED DATA.

SET     NA_OTHER          ;NO ADDITIONAL DATA.

CALL    TEST1E_ERROR      ;PRINT AN ERROR MESSAGE.

ENDIF

LDA     ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC      1$                ;...JUMP TO BEGINNING OF ERROR LOOP.

MVI     A,2               ;SECOND PART OF TEST 1E.
STA     CON_ERR_NUM

LHLD    TEST1E_DAT        ;MOVE THE COUNTER MODE EXPECTED DATA
SHLD    EXP_DAT           ; INTO EXP_DAT.

CALL    CLR_CG_PNT        ;CLEAR THE COUNTER GROUP POINTER.

DO      5

CALL    INC_CG_PNT        ;INCREMENT THE COUNTER GROUP POINTER.

CALL    MASTER_RESET_RESTORE      ;PERFORM A MASTER RESET AND
                          ; RESTORE MASTER REG TO NORMAL VALUE

CALL    READ_MODE         ;READ THE MODE REGISTER.

IFNEQ   EXP_DAT,REC_DAT,2         ;COMPARE THE EXPECTED

                                  ; AND RECEIVED DATA.

CALL    TEST1E_ERROR      ;PRINT AN ERROR MESSAGE.

ENDIF

LDA     ERROR_CON         ;IF THE ERROR LOOPING FLAG IS SET...
RRC
JC      2$                ;...JUMP TO BEGINNING OF ERROR LOOP.

ENDDO

CALL    CHECK_RD          ;CALL BOOT CODE IF UNDER RD CONTROL

MVI     A,3               ;THIRD PART OF TEST 1E.
```

```
73A3 1290 0000'32                                  STA     CON_ERR_NUM
73A6 1291
73A6 1292 000E'2A                                  LHLD    ZERO_DAT        ;MOVE THE COUNTER LOAD EXPECTED DATA
73A9 1293 0028'22                                  SHLD    EXP_DAT         ; INTO EXP_DAT.
73AC 1294
73AC 1295 004D'CD                                  CALL    CLR_CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.
73AF 1296
73AF 1297 C5 46 00'00'21      ******************   DO      5
          77 05 3E            *
73B7 1298                     *
73B7 1299 0052'CD             *                    CALL    INC_CG_PNT      ;INCREMENT THE COUNTER GROUP POINTER.
73BA 1300                     *
73BA 1301 0042'CD             *          3$:       CALL    MASTER_RESET_RESTORE    ;PERFORM A MASTER RESET AND
73BD 1302                     *                                            ; RESTORE MASTER REG TO NORMAL VALUE
73BD 1303                     *
73BD 1304 0091'CD             *                    CALL    READ_LOAD       ;READ THE LOAD REGISTER.
73C0 1305                     *
73C0 1306 00'28'21            * ****************   IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED
          02 0E 00'2B'11      * *
          03D1'CA 0000'CD     * *
73CE 1307                     * *                                                   ; AND RECEIVED DATA.
73CE 1308                     * *
73CE 1309 0168'CD             * *                  CALL    TEST1E_ERROR    ;PRINT AN ERROR MESSAGE.
73D1 1310                     * *
73D1 1311                     * ****************   ENDIF
73D1 1312                     *
73D1 1313 0000'3A             *                    LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
73D4 1314 0F                  *                    RRC
73D5 1315 03BA'DA             *                    JC      3$              ;...JUMP TO BEGINNING OF ERROR LOOP.
73D8 1316                     *
73D8 1317 35 00'00'21         ******************   ENDDO
          70 C1 03B7'C2
73E1 1318
73E1 1319
73E1 1320 002D'CD                                  CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
73E4 1321
73E4 1322 04 3E                                    MVI     A,4             ;FOURTH PART OF TEST 1E.
73E6 1323 0000'32                                  STA     CON_ERR_NUM
73E9 1324
73E9 1325 000E'2A                                  LHLD    ZERO_DAT        ;MOVE THE COUNTER HOLD EXPECTED DATA
73EC 1326 0028'22                                  SHLD    EXP_DAT         ; INTO EXP_DAT.
73EF 1327
73EF 1328 004D'CD                                  CALL    CLR_CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.
73F2 1329
73F2 1330 C5 46 00'00'21      ******************   DO      5
          77 05 3E            *
73FA 1331                     *
73FA 1332 0052'CD             *                    CALL    INC_CG_PNT      ;INCREMENT THE COUNTER GROUP POINTER.
73FD 1333                     *
73FD 1334 0042'CD             *          4$:       CALL    MASTER_RESET_RESTORE    ;PERFORM A MASTER RESET AND
7400 1335                     *                                            ; RESTORE MASTER REG TO NORMAL VALUE
7400 1336                     *
7400 1337 00A9'CD             *                    CALL    READ_HOLD       ;READ THE HOLD REGISTER.
7403 1338                     *
7403 1339 00'28'21            * ****************   IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED
          02 0E 00'2B'11      * *
          0414'CA 0000'CD     * *
7411 1340                     * *                                                   ; AND RECEIVED DATA.
7411 1341                     * *
7411 1342 0168'CD             * *                  CALL    TEST1E_ERROR    ;PRINT AN ERROR MESSAGE.
```

```
7414 1343                    * *
7414 1344                    * ****************          ENDIF
7414 1345                    *
7414 1346 0000'3A            *                           LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7417 1347 0F                 *                           RRC
7418 1348 03FD'DA            *                           JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
741B 1349                    *
741B 1350 35 00'00'21        ******************          ENDDO
          70 C1 03FA'C2
7424 1351
7424 1352
7424 1353 3C D3                                          TAIL
7426 1354
7426 1355
7426 1356                                   ;*****************************************************************************
7426 1357                                   ;
7426 1358                                   ;
7426 1359                                   ;       TEST1F          MASTER MODE REGISTER TEST
7426 1360                                   ;       ------          -------------------------
7426 1361                                   ;
7426 1362                                   ;
7426 1363                                   ;               THIS TEST LOADS THE MASTER MODE REGISTER WITH DATA PATTERNS
7426 1364                                   ;       CONSISTING OF ALL ZEROS, ALL ONES, A ONE SHIFTED THROUGH A ZERO FIELD,
7426 1365                                   ;       AND A ZERO THROUGH A ONE FIELD. THE DATA SENT IS COMPARED TO THE DATA
7426 1366                                   ;       RECEIVED TO EVALUATE THE REGISTER'S INTEGRITY. BITS 13 AND 14 ARE
7426 1367                                   ;       ALWAYS LEFT AT ZERO IN THIS TEST AS THEY CONTROL THE DATA BUS WIDTH
7426 1368                                   ;       AND DATA POINTER INCREMENTING.
7426 1369                                   ;
7426 1370                                   ;
7426 1371                                   ;       TEST STEPS:
7426 1372                                   ;
7426 1373                                   ;               1) WRITE HEX FF (MASTER RESET) TO COMMAND REGISTER.
7426 1374                                   ;               2) DISABLE DATA POINTER SEQUENCING.
7426 1375                                   ;               3) WRITE DATA PATTERN TO MASTER MODE REGISTER.
7426 1376                                   ;               4) READ DATA FROM MASTER MODE REGISTER.
7426 1377                                   ;               5) COMPARE SENT AND RECEIVED DATA.
7426 1378                                   ;               6) PRINT ERROR IF ANY.
7426 1379                                   ;               7) REPEAT FOR ALL DATA PATTERNS.
7426 1380                                   ;
7426 1381                                   ;
7426 1382                                   ;       ASSUMPTIONS: THE PREVIOUS INTERVAL TIMER TEST HAS BEEN RUN
7426 1383                                   ;                    SUCCESSFULLY.
7426 1384                                   ;
7426 1385                                   ;
7426 1386                                   ;       ERROR1: MASTER MODE REGISTER ERROR.
7426 1387                                   ;
7426 1388                                   ;
7426 1389                                   ;       DEBUG: SINCE ALL THE EXTERNAL SIGNALS USED PERFORM THIS TEST
7426 1390                                   ;              WERE USED IN THE PREVIOUS TEST, IF THIS TEST FAILS,
7426 1391                                   ;              SUSPECT THE INTERVAL TIMER CHIP.
7426 1392                                   ;
7426 1393                                   ;
7426 1394                                   ;*****************************************************************************
7426 1395
7426 1396
7426 1397
7426 1398
7426 1399 0000'CD 1F 3E                     TEST1F:         HEAD    <^X1F>
          0F 0000'3A
          C9 0433'D2
```

```
          0F 0000'3A
          3D D3 052E'DA
743C 1400
743C 1401 003D'CD                                   CALL    MASTER_RESET     ;PERFORM A MASTER RESET.
743F 1402
743F 1403 000E'2A                                   LHLD    ZERO_DAT         ;LOAD ALL ZEROS INTO TEMP_DAT.
7442 1404 0025'22                                   SHLD    TEMP_DAT
7445 1405
7445 1406 00ED'CD                                   CALL    MM_BITS          ;SET BITS: MM14=1, MM13=0, AND PUT
7448 1407                                                                    ; DATA IN EXP_DAT.
7448 1408 00'28'21                        1$:       LXI     H,EXP_DAT        ;WRITE EXP_DAT TO THE MASTER MODE
744B 1409 0057'CD                                   CALL    WRITE_MASTER     ; REGISTER.
744E 1410
744E 1411 005F'CD                                   CALL    READ_MASTER      ;READ THE MASTER MODE REGISTER.
7451 1412
7451 1413 00'28'21          ******************      IFNEQ   EXP_DAT,REC_DAT,2        ;COMPARE EXPECTED AND
          02 0E 00'2B'11    *
          0462'CA 0000'CD   *
745F 1414                   *                                                        ; RECEIVED DATA.
745F 1415                   *
745F 1416 01A2'CD           *                       CALL    TEST1F_ERROR     ;PRINT AN ERROR MESSAGE.
7462 1417                   *
7462 1418                   ******************      ENDIF
7462 1419
7462 1420 0000'3A                                   LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
7465 1421 0F                                        RRC
7466 1422 0448'DA                                   JC      1$               ;...JUMP TO BEGINNING OF ERROR LOOP.
7469 1423
7469 1424
7469 1425
7469 1426 003D'CD                                   CALL    MASTER_RESET     ;PERFORM A MASTER RESET.
746C 1427
746C 1428 0011'2A                                   LHLD    ONE_DAT          ;LOAD ALL ONES INTO TEMP_DAT.
746F 1429 0025'22                                   SHLD    TEMP_DAT
7472 1430
7472 1431 00ED'CD                                   CALL    MM_BITS          ;SET BITS: MM14=1, MM13=0, AND PUT
7475 1432                                                                    ; DATA IN EXP_DAT.
7475 1433
7475 1434 00'28'21                        2$:       LXI     H,EXP_DAT        ;WRITE EXP_DAT TO THE MASTER MODE
7478 1435 0057'CD                                   CALL    WRITE_MASTER     ; REGISTER.
747B 1436
747B 1437 005F'CD                                   CALL    READ_MASTER      ;READ THE MASTER MODE REGISTER.
747E 1438
747E 1439 00'28'21          ******************      IFNEQ   EXP_DAT,REC_DAT,2        ;COMPARE EXPECTED AND
          02 0E 00'2B'11    *
          048F'CA 0000'CD   *
748C 1440                   *                                                        ; RECEIVED DATA.
748C 1441                   *
748C 1442 01A2'CD           *                       CALL    TEST1F_ERROR     ;PRINT AN ERROR MESSAGE.
748F 1443                   *
748F 1444                   ******************      ENDIF
748F 1445
748F 1446 0000'3A                                   LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
7492 1447 0F                                        RRC
7493 1448 0475'DA                                   JC      2$               ;...JUMP TO BEGINNING OF ERROR LOOP.
7496 1449
7496 1450
7496 1451 002D'CD                                   CALL    CHECK_RD         ;CALL BOOT CODE IF UNDER RD CONTROL
7499 1452
7499 1453 003D'CD                                   CALL    MASTER_RESET     ;PERFORM A MASTER RESET.
```

```
749C 1454
749C 1455 000F'2A                                  LHLD    ONE_SHIFT        ;LOAD DATA FOR A ONE SHIFTED THROUGH
749F 1456 0025'22                                  SHLD    TEMP_DAT         ; A ZERO FIELD INTO TEMP_DAT.
74A2 1457
74A2 1458 C5 46 00'00'21      ******************   DO      16
          77 10 3E            *
74AA 1459                     *
74AA 1460 00ED'CD             *                    CALL    MM_BITS          ;SET BITS: MM14=1, MM13=0, AND PUT
74AD 1461                     *                                             ; DATA IN EXP_DAT.
74AD 1462                     *
74AD 1463 00'28'21            *            3$:     LXI     H,EXP_DAT        ;WRITE EXP_DAT TO THE MASTER MODE
74B0 1464 0057'CD             *                    CALL    WRITE_MASTER     ; REGISTER.
74B3 1465                     *
74B3 1466 005F'CD             *                    CALL    READ_MASTER      ;READ THE MASTER MODE REGISTER.
74B6 1467                     *
74B6 1468 00'28'21            * ****************   IFNEQ   EXP_DAT,REC_DAT,2        ;COMPARE EXPECTED AND
          02 0E 00'2B'11      * *
          04C7'CA 0000'CD     * *
74C4 1469                     * *                                                  ; RECEIVED DATA.
74C4 1470                     * *
74C4 1471 01A2'CD             * *                  CALL    TEST1F_ERROR     ;PRINT AN ERROR MESSAGE.
74C7 1472                     * *
74C7 1473                     * ****************   ENDIF
74C7 1474                     *
74C7 1475 0000'3A             *                    LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
74CA 1476 0F                  *                    RRC
74CB 1477 04AD'DA             *                    JC      3$               ;...JUMP TO BEGINNING OF ERROR LOOP.
74CE 1478                     *
74CE 1479                     *
74CE 1480 B7                  *                    ORA     A                ;CLEAR THE CARRY.
74CF 1481                     *
74CF 1482 00'24'21            *                    LXI     H,TEMP_SHIFT     ;SHIFT THE ONE THROUGH THE ZERO FIELD.
74D2 1483 0000'22             *                    SHLD    SHIFT_PNT
74D5 1484 0000'CD             *                    CALL    SHIFTER
74D8 1485                     *
74D8 1486 35 00'00'21         ******************   ENDDO
          70 C1 04AA'C2
74E1 1487
74E1 1488
74E1 1489 002D'CD                                  CALL    CHECK_RD         ;CALL BOOT CODE IF UNDER RD CONTROL
74E4 1490
74E4 1491 003D'CD                                  CALL    MASTER_RESET     ;PERFORM A MASTER RESET.
74E7 1492
74E7 1493 0012'2A                                  LHLD    ZERO_SHIFT       ;LOAD DATA FOR A ZERO SHIFTED THROUGH
74EA 1494 0025'22                                  SHLD    TEMP_DAT         ; A ONE FIELD INTO TEMP_DAT.
74ED 1495
74ED 1496 C5 46 00'00'21      ******************   DO      16
          77 10 3E            *
74F5 1497                     *
74F5 1498 00ED'CD             *                    CALL    MM_BITS          ;SET BITS: MM14=1, MM13=0, AND PUT
74F8 1499                     *                                             ; DATA IN EXP_DAT.
74F8 1500                     *
74F8 1501 00'28'21            *            4$:     LXI     H,EXP_DAT        ;WRITE EXP_DAT TO THE MASTER MODE
74FB 1502 0057'CD             *                    CALL    WRITE_MASTER     ; REGISTER.
74FE 1503                     *
74FE 1504 005F'CD             *                    CALL    READ_MASTER      ;READ THE MASTER MODE REGISTER.
7501 1505                     *
7501 1506 00'28'21            * ****************   IFNEQ   EXP_DAT,REC_DAT,2        ;COMPARE EXPECTED AND
          02 0E 00'2B'11      * *
          0512'CA 0000'CD     * *
```

```
750F 1507                    * *                                  ; RECEIVED DATA.
750F 1508                    * *
750F 1509 01A2'CD            * *                    CALL    TEST1F_ERROR    ;PRINT AN ERROR MESSAGE.
7512 1510                    * *
7512 1511                    * ****************      ENDIF
7512 1512                    *
7512 1513 0000'3A            *                      LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7515 1514 0F                 *                      RRC
7516 1515 04F8'DA            *                      JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7519 1516                    *
7519 1517                    *
7519 1518 37                 *                      STC     A               ;SET THE CARRY.
751A 1519                    *
751A 1520 00'24'21           *                      LXI     H,TEMP_SHIFT    ;SHIFT THE ZERO THROUGH THE ONE FIELD.
751D 1521 0000'22            *                      SHLD    SHIFT_PNT
7520 1522 0000'CD            *                      CALL    SHIFTER
7523 1523                    *
7523 1524 35 00'00'21        ******************      ENDDO
          70 C1 04F5'C2
752C 1525
752C 1526
752C 1527 3C D3                                     TAIL
752E 1528
752E 1529
752E 1530                    ;*****************************************************************************
752E 1531                    ;
752E 1532                    ;
752E 1533                    ;       TEST20          ELEMENT CYCLE REGISTER TEST
752E 1534                    ;       ------          ---------------------------
752E 1535                    ;
752E 1536                    ;
752E 1537                    ;               THIS TEST LOADS THE MODE, HOLD AND LOAD REGISTERS IN EACH
752E 1538                    ;       COUNTER GROUP WITH DATA PATTERNS CONSISTING OF ALL ZEROS, ALL ONES,
752E 1539                    ;       A ONE SHIFTED THROUGH A ZERO FIELD, AND A ZERO SHIFTED THROUGH A ONE
752E 1540                    ;       FIELD. THE DATA IS THEN READ BACK AND COMPARED WITH THE DATA SENT.
752E 1541                    ;
752E 1542                    ;
752E 1543                    ;       TEST STEPS:
752E 1544                    ;
752E 1545                    ;               1) PERFORM MASTER RESET.
752E 1546                    ;               2) POINT TO CG1 MODE REGISTER.
752E 1547                    ;               3) WRITE DATA PATTERN TO MODE REGISTER.
752E 1548                    ;               4) READ DATA PATTERN FROM MODE REGISTER.
752E 1549                    ;               5) COMPARE SENT AND RECEIVED DATA.
752E 1550                    ;               6) PRINT ERRORS IF ANY.
752E 1551                    ;               7) REPEAT FOR ALL DATA PATTERNS.
752E 1552                    ;               8) REPEAT FOR THE OTHER 14 REGISTERS IN ELEMENT CYCLE.
752E 1553                    ;
752E 1554                    ;
752E 1555                    ;       ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.
752E 1556                    ;
752E 1557                    ;
752E 1558                    ;       ERROR1: ELEMENT CYCLE REGISTER ERROR - COUNTER MODE REGISTER.
752E 1559                    ;       ERROR2: ELEMENT CYCLE REGISTER ERROR - COUNTER LOAD REGISTER.
752E 1560                    ;       ERROR3: ELEMENT CYCLE REGISTER ERROR - COUNTER HOLD REGISTER.
752E 1561                    ;
752E 1562                    ;       NOTE: ADDITIONAL DATA WILL SPECIFY COUNTER GROUP.
752E 1563                    ;
752E 1564                    ;
752E 1565                    ;       DEBUG: SUSPECT INTERVAL TIMER CHIP.
```

```
752E 1566                                   ;
752E 1567                                   ;
752E 1568                                   ;*****************************************************************************
752E 1569
752E 1570
752E 1571
752E 1572
752E 1573 0000'CD 20 3E                     TEST20:         HEAD    <^X20>
          0F 0000'3A
          C9 053B'D2
          0F 0000'3A
          3D D3 06AA'DA
7544 1574
7544 1575 0042'CD                                           CALL    MASTER_RESET_RESTORE    ;PERFORM A MASTER RESET AND
7547 1576                                                                                   ; RESTORE MASTER REG TO NORMAL VALUE
7547 1577
7547 1578 004D'CD                                           CALL    CLR_CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.
754A 1579
754A 1580 C5 46 00'00'21        ******************          DO      5
          77 05 3E              *
7552 1581                       *
7552 1582 0052'CD               *                           CALL    INC_CG_PNT      ;INCREMENT THE COUNTER GROUP POINTER.
7555 1583                       *
7555 1584 000E'2A               *                           LHLD    ZERO_DAT        ;LOAD ALL ZEROS INTO EXP_DAT.
7558 1585 0028'22               *                           SHLD    EXP_DAT
755B 1586                       *
755B 1587 02A7'CD               *                           CALL    TEST20_ONES_ZEROS ;CHECK REGISTERS WITH ALL ZEROS DATA
755E 1588                       *
755E 1589 0011'2A               *                           LHLD    ONE_DAT         ;LOAD ALL ONES INTO EXP_DAT.
7561 1590 0028'22               *                           SHLD    EXP_DAT
7564 1591                       *
7564 1592 02A7'CD               *                           CALL    TEST20_ONES_ZEROS ;CHECK REGISTERS WITH ALL ONES DATA
7567 1593                       *
7567 1594                       *
7567 1595 000F'2A               *                           LHLD    ONE_SHIFT       ;LOAD A PATTERN TO SHIFT A ONE THROUGH
756A 1596 0025'22               *                           SHLD    TEMP_DAT        ; A ZERO FIELD INTO TEMP_DAT.
756D 1597                       *
756D 1598 C5 46 00'00'21        * ****************          DO      16
          77 10 3E              * *
7575 1599                       * *
7575 1600 0025'2A               * *                         LHLD    TEMP_DAT        ;MOVE TEMP_DAT TO EXP_DAT.
7578 1601 0028'22               * *                         SHLD    EXP_DAT
757B 1602                       * *
757B 1603 00'28'21              * *                 7$:     LXI     H,EXP_DAT       ;WRITE EXP_DAT TO THE COUNTER MODE
757E 1604 006D'CD               * *                         CALL    WRITE_MODE      ; REGISTER.
7581 1605                       * *
7581 1606 0079'CD               * *                         CALL    READ_MODE       ;READ THE COUNTER MODE REGISTER.
7584 1607                       * *
7584 1608 00'28'21              * * **************          IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11        * * *
          059A'CA 0000'CD       * * *
7592 1609                       * * *                                                       ; RECEIVED DATA.
7592 1610                       * * *
7592 1611 01 3E                 * * *                       MVI     A,1             ;ERROR NUMBER 1.
7594 1612 0000'32               * * *                       STA     CON_ERR_NUM
7597 1613                       * * *
7597 1614 0168'CD               * * *                       CALL    TEST1E_ERROR    ;PRINT AN ERROR MESSAGE.
759A 1615                       * * *
759A 1616                       * * **************          ENDIF
759A 1617                       * *
```

```
759A 1618 0000'3A          * *                  LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
759D 1619 0F               * *                  RRC
759E 1620 057B'DA          * *                  JC      7$               ;...JUMP TO BEGINNING OF ERROR LOOP.
75A1 1621                  * *
75A1 1622 002D'CD          * *                  CALL    CHECK_RD         ;CALL BOOT CODE IF UNDER RD CONTROL
75A4 1623                  * *
75A4 1624 00'28'21         * *          8$:     LXI     H,EXP_DAT        ;WRITE EXP_DAT TO THE COUNTER LOAD
75A7 1625 0085'CD          * *                  CALL    WRITE_LOAD       ; REGISTER.
75AA 1626                  * *
75AA 1627 0091'CD          * *                  CALL    READ_LOAD        ;READ THE COUNTER LOAD REGISTER.
75AD 1628                  * *
75AD 1629 00'28'21         * * **************   IFNEQ   EXP_DAT,REC_DAT,2        ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11   * * *
          05C3'CA 0000'CD  * * *
75BB 1630                  * * *                                            ; RECEIVED DATA.
75BB 1631                  * * *
75BB 1632 02 3E            * * *                MVI     A,2              ;ERROR NUMBER 2.
75BD 1633 0000'32          * * *                STA     CON_ERR_NUM
75C0 1634                  * * *
75C0 1635 0168'CD          * * *                CALL    TEST1E_ERROR     ;PRINT AN ERROR MESSAGE.
75C3 1636                  * * *
75C3 1637                  * * **************   ENDIF
75C3 1638                  * *
75C3 1639 0000'3A          * *                  LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
75C6 1640 0F               * *                  RRC
75C7 1641 05A4'DA          * *                  JC      8$               ;...JUMP TO BEGINNING OF ERROR LOOP.
75CA 1642                  * *
75CA 1643                  * *
75CA 1644 00'28'21         * *          9$:     LXI     H,EXP_DAT        ;WRITE EXP_DAT TO THE COUNTER HOLD
75CD 1645 009D'CD          * *                  CALL    WRITE_HOLD       ; REGISTER.
75D0 1646                  * *
75D0 1647 00A9'CD          * *                  CALL    READ_HOLD        ;READ THE COUNTER HOLD REGISTER.
75D3 1648                  * *
75D3 1649 00'28'21         * * **************   IFNEQ   EXP_DAT,REC_DAT,2        ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11   * * *
          05E9'CA 0000'CD  * * *
75E1 1650                  * * *                                            ; RECEIVED DATA.
75E1 1651                  * * *
75E1 1652 03 3E            * * *                MVI     A,3              ;ERROR NUMBER 3.
75E3 1653 0000'32          * * *                STA     CON_ERR_NUM
75E6 1654                  * * *
75E6 1655 0168'CD          * * *                CALL    TEST1E_ERROR     ;PRINT AN ERROR MESSAGE.
75E9 1656                  * * *
75E9 1657                  * * **************   ENDIF
75E9 1658                  * *
75E9 1659 0000'3A          * *                  LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
75EC 1660 0F               * *                  RRC
75ED 1661 05CA'DA          * *                  JC      9$               ;...JUMP TO BEGINNING OF ERROR LOOP.
75F0 1662                  * *
75F0 1663 B7               * *                  ORA     A                ;CLEAR THE CARRY.
75F1 1664                  * *
75F1 1665 00'24'21         * *                  LXI     H,TEMP_SHIFT     ;SHIFT THE ONE THROUGH THE ZERO FIELD.
75F4 1666 0000'22          * *                  SHLD    SHIFT_PNT
75F7 1667 0000'CD          * *                  CALL    SHIFTER
75FA 1668                  * *
75FA 1669 35 00'00'21      * ***************    ENDDO
          70 C1 0575'C2    *
7603 1670                  *
7603 1671 0012'2A          *                    LHLD    ZERO_SHIFT       ;LOAD A PATTERN TO SHIFT A ZERO THROUGH
7606 1672 0025'22          *                    SHLD    TEMP_DAT         ; A ONE FIELD INTO TEMP_DAT.
```

```
7609 1673                    *
7609 1674 C5 46 00'00'21     * ****************          DO      16
          77 10 3E           * *
7611 1675                    * *
7611 1676 0025'2A            * *                         LHLD    TEMP_DAT        ;MOVE TEMP_DAT TO EXP_DAT.
7614 1677 0028'22            * *                         SHLD    EXP_DAT
7617 1678                    * *
7617 1679 00'28'21           * *                 10$:    LXI     H,EXP_DAT       ;WRITE EXP_DAT TO THE COUNTER MODE
761A 1680 006D'CD            * *                         CALL    WRITE_MODE      ; REGISTER.
761D 1681                    * *
761D 1682 0079'CD            * *                         CALL    READ_MODE       ;READ THE COUNTER MODE REGISTER.
7620 1683                    * *
7620 1684 00'28'21           * * **************          IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11     * * *
          0636'CA 0000'CD    * * *
762E 1685                    * * *                                                       ; RECEIVED DATA.
762E 1686                    * * *
762E 1687 01 3E              * * *                       MVI     A,1             ;ERROR NUMBER 1.
7630 1688 0000'32            * * *                       STA     CON_ERR_NUM
7633 1689                    * * *
7633 1690 0168'CD            * * *                       CALL    TEST1E_ERROR    ;PRINT AN ERROR MESSAGE.
7636 1691                    * * *
7636 1692                    * * **************          ENDIF
7636 1693                    * *
7636 1694 0000'3A            * *                         LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7639 1695 0F                 * *                         RRC
763A 1696 0617'DA            * *                         JC      10$             ;...JUMP TO BEGINNING OF ERROR LOOP.
763D 1697                    * *
763D 1698 002D'CD            * *                         CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7640 1699                    * *
7640 1700 00'28'21           * *                 11$:    LXI     H,EXP_DAT       ;WRITE EXP_DAT TO THE COUNTER LOAD
7643 1701 0085'CD            * *                         CALL    WRITE_LOAD      ; REGISTER.
7646 1702                    * *
7646 1703 0091'CD            * *                         CALL    READ_LOAD       ;READ THE COUNTER LOAD REGISTER.
7649 1704                    * *
7649 1705 00'28'21           * * **************          IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11     * * *
          065F'CA 0000'CD    * * *
7657 1706                    * * *                                                       ; RECEIVED DATA.
7657 1707                    * * *
7657 1708 02 3E              * * *                       MVI     A,2             ;ERROR NUMBER 2.
7659 1709 0000'32            * * *                       STA     CON_ERR_NUM
765C 1710                    * * *
765C 1711 0168'CD            * * *                       CALL    TEST1E_ERROR    ;PRINT AN ERROR MESSAGE.
765F 1712                    * * *
765F 1713                    * * **************          ENDIF
765F 1714                    * *
765F 1715 0000'3A            * *                         LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7662 1716 0F                 * *                         RRC
7663 1717 0640'DA            * *                         JC      11$             ;...JUMP TO BEGINNING OF ERROR LOOP.
7666 1718                    * *
7666 1719                    * *
7666 1720 00'28'21           * *                 12$:    LXI     H,EXP_DAT       ;WRITE EXP_DAT TO THE COUNTER HOLD
7669 1721 009D'CD            * *                         CALL    WRITE_HOLD      ; REGISTER.
766C 1722                    * *
766C 1723 00A9'CD            * *                         CALL    READ_HOLD       ;READ THE COUNTER HOLD REGISTER.
766F 1724                    * *
766F 1725 00'28'21           * * **************          IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11     * * *
          0685'CA 0000'CD    * * *
```

```
767D 1726                    * * *                                                  ; RECEIVED DATA.
767D 1727                    * * *
767D 1728 03 3E              * * *                          MVI     A,3             ;ERROR NUMBER 3.
767F 1729 0000'32            * * *                          STA     CON_ERR_NUM
7682 1730                    * * *
7682 1731 0168'CD            * * *                          CALL    TEST1E_ERROR    ;PRINT AN ERROR MESSAGE.
7685 1732                    * * *
7685 1733                    * * **************             ENDIF
7685 1734                    * *
7685 1735 0000'3A            * *                            LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7688 1736 0F                 * *                            RRC
7689 1737 0666'DA            * *                            JC      12$             ;...JUMP TO BEGINNING OF ERROR LOOP.
768C 1738                    * *
768C 1739 37                 * *                            STC                     ;SET THE CARRY.
768D 1740                    * *
768D 1741 00'24'21           * *                            LXI     H,TEMP_SHIFT    ;SHIFT THE ONE THROUGH THE ZERO FIELD.
7690 1742 0000'22            * *                            SHLD    SHIFT_PNT
7693 1743 0000'CD            * *                            CALL    SHIFTER
7696 1744                    * *
7696 1745 35 00'00'21        * ****************             ENDDO
          70 C1 0611'C2      *
769F 1746                    *
769F 1747 35 00'00'21        ******************             ENDDO
          70 C1 0552'C2
76A8 1748
76A8 1749
76A8 1750 3C D3                                             TAIL
76AA 1751
76AA 1752
76AA 1753                 ;*****************************************************************************
76AA 1754
76AA 1755
76AA 1756                 ;     TEST21          ALARM REGISTER TEST
76AA 1757                 ;     ------          -------------------
76AA 1758
76AA 1759
76AA 1760                 ;             THIS TEST LOADS THE TWO ALARM REGISTERS WITH DATA PATTERNS
76AA 1761                 ;     CONSISTING OF ALL ZEROS, ALL ONES, A ONE SHIFTED THROUGH A ZERO FIELD,
76AA 1762                 ;     AND A ZERO THROUGH A ONE FIELD. THE DATA IS THEN READ BACK AND COMPARED
76AA 1763                 ;     WITH THE DATA SENT.
76AA 1764
76AA 1765
76AA 1766                 ;     TEST STEPS:
76AA 1767
76AA 1768                 ;             1) PERFORM MASTER RESET.
76AA 1769                 ;             2) POINT TO ALARM REGISTER 1.
76AA 1770                 ;             3) WRITE DATA PATTERN TO ALARM REGISTER 1.
76AA 1771                 ;             4) READ DATA PATTERN FROM ALARM REGISTER 1.
76AA 1772                 ;             5) COMPARE SENT AND RECEIVED DATA AND PRINT ERROR IF ANY.
76AA 1773                 ;             6) REPEAT FOR ALL DATA PATTERNS.
76AA 1774                 ;             7) REPEAT FOR ALARM REGISTER 2.
76AA 1775
76AA 1776
76AA 1777                 ;     ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.
76AA 1778
76AA 1779
76AA 1780                 ;     ERROR1: ALARM REGISTER 1 FAILURE.
76AA 1781                 ;     ERROR2: ALARM REGISTER 2 FAILURE.
76AA 1782
76AA 1783
```

```
76AA 1784                                   ;       DEBUG: SUSPECT INTERVAL TIMER CHIP.
76AA 1785                                   ;
76AA 1786                                   ;
76AA 1787                                   ;*****************************************************************************
76AA 1788
76AA 1789
76AA 1790
76AA 1791 0000'CD 21 3E                     TEST21:         HEAD    <^X21>
          0F 0000'3A
          C9 06B7'D2
          0F 0000'3A
          3D D3 07BD'DA
76C0 1792
76C0 1793 0042'CD                                           CALL    MASTER_RESET_RESTORE    ;PERFORM A MASTER RESET AND
76C3 1794                                                                                   ; RESTORE MASTER REG TO NORMAL VALUE
76C3 1795
76C3 1796 004D'CD                                           CALL    CLR_CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.
76C6 1797
76C6 1798 C5 46 00'00'21        ******************          DO      2
          77 02 3E              *
76CE 1799                       *
76CE 1800 0052'CD               *                           CALL    INC_CG_PNT      ;INCREMENT THE COUNTER GROUP POINTER.
76D1 1801                       *
76D1 1802 000E'2A               *                           LHLD    ZERO_DAT        ;MOVE ALL ZEROS TO EXP_DAT.
76D4 1803 0028'22               *                           SHLD    EXP_DAT
76D7 1804                       *
76D7 1805 00'28'21              *                   1$:     LXI     H,EXP_DAT       ;WRITE ALL ZEROS TO THE ALARM REG.
76DA 1806 00B5'CD               *                           CALL    WRITE_ALARM
76DD 1807                       *
76DD 1808 00C7'CD               *                           CALL    READ_ALARM      ;READ THE ALARM REGISTER.
76E0 1809                       *
76E0 1810 00'28'21              * ****************          IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11        * *
          06F1'CA 0000'CD       * *
76EE 1811                       * *                                                         ; RECEIVED DATA.
76EE 1812                       * *
76EE 1813 01CE'CD               * *                         CALL    TEST21_ERROR    ;PRINT AN ERROR MESSAGE.
76F1 1814                       * *
76F1 1815                       * ****************          ENDIF
76F1 1816                       *
76F1 1817 0000'3A               *                           LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
76F4 1818 0F                    *                           RRC
76F5 1819 06D7'DA               *                           JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
76F8 1820                       *
76F8 1821                       *
76F8 1822 0011'2A               *                           LHLD    ONE_DAT         ;MOVE ALL ONES TO EXP_DAT.
76FB 1823 0028'22               *                           SHLD    EXP_DAT
76FE 1824                       *
76FE 1825 00'28'21              *                   2$:     LXI     H,EXP_DAT       ;WRITE ALL ONES TO THE ALARM REG.
7701 1826 00B5'CD               *                           CALL    WRITE_ALARM
7704 1827                       *
7704 1828 00C7'CD               *                           CALL    READ_ALARM      ;READ THE ALARM REGISTER.
7707 1829                       *
7707 1830 00'28'21              * ****************          IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11        * *
          0718'CA 0000'CD       * *
7715 1831                       * *                                                         ; RECEIVED DATA.
7715 1832                       * *
7715 1833 01CE'CD               * *                         CALL    TEST21_ERROR    ;PRINT AN ERROR MESSAGE.
7718 1834                       * *
```

```
7718 1835                    * ****************          ENDIF                                                    7B
7718 1836                    *                                                                                    7B
7718 1837 0000'3A            *                           LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...  7B
771B 1838 0F                 *                           RRC                                                      7B
771C 1839 06FE'DA            *                           JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.  7B
771F 1840                    *                                                                                    7B
771F 1841                    *                                                                                    7B
771F 1842                    *                                                                                    7B
771F 1843 000F'2A            *                           LHLD    ONE_SHIFT       ;LOAD A ONE SHIFTED THROUGH ZERO      7B
7722 1844 0025'22            *                           SHLD    TEMP_DAT        ; STRING INTO TEMP_DAT.               7B
7725 1845                    *                                                                                    7B
7725 1846 C5 46 00'00'21     * ****************          DO      16                                                    7B
          77 10 3E           * *                                                                                  7B
772D 1847                    * *                                                                                  7B
772D 1848 0025'2A            * *                         LHLD    TEMP_DAT        ;LOAD A DATA PATTERN INTO EXP_DAT.    7B
7730 1849 0028'22            * *                         SHLD    EXP_DAT                                               7B
7733 1850                    * *                                                                                  7B
7733 1851 00'28'21           * *                 3$:     LXI     H,EXP_DAT       ;WRITE A DATA PATTERN TO THE ALARM REG. 7B
7736 1852 00B5'CD            * *                         CALL    WRITE_ALARM                                           7B
7739 1853                    * *                                                                                  7B
7739 1854 00C7'CD            * *                         CALL    READ_ALARM      ;READ THE ALARM REGISTER.             7B
773C 1855                    * *                                                                                  7B
773C 1856 00'28'21           * * **************          IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND 7B
          02 0E 00'2B'11     * * *                                                                                7B
          074D'CA 0000'CD    * * *                                                                                7B
774A 1857                    * * *                                                   ; RECEIVED DATA.              7B
774A 1858                    * * *                                                                                7B
774A 1859 01CE'CD            * * *                       CALL    TEST21_ERROR    ;PRINT AN ERROR MESSAGE.              7B
774D 1860                    * * *                                                                                7B
774D 1861                    * * **************          ENDIF                                                    7B
774D 1862                    * *                                                                                  7B
774D 1863 0000'3A            * *                         LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...  7B
7750 1864 0F                 * *                         RRC                                                      7B
7751 1865 0733'DA            * *                         JC      3$              ;...JUMP TO BEGINNING OF ERROR LOOP.  7B
7754 1866                    * *                                                                                  7B
7754 1867 B7                 * *                         ORA     A               ;CLEAR THE CARRY.                     7B
7755 1868                    * *                                                                                  7B
7755 1869 00'24'21           * *                         LXI     H,TEMP_SHIFT    ;SHIFT THE ONE THROUGH THE ZERO FIELD 7B
7758 1870 0000'22            * *                         SHLD    SHIFT_PNT                                             7B
775B 1871 0000'CD            * *                         CALL    SHIFTER                                               7B
775E 1872                    * *                                                                                  7B
775E 1873 35 00'00'21        * ****************          ENDDO
          70 C1 072D'C2      *
7767 1874                    *
7767 1875 002D'CD            *                           CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL   7B
776A 1876                    *                                                                                    7B
776A 1877 0012'2A            *                           LHLD    ZERO_SHIFT      ;LOAD A ZERO SHIFTED THROUGH ONE STRING 7B
776D 1878 0025'22            *                           SHLD    TEMP_DAT        ; INTO TEMP_DAT.                      7B
7770 1879                    *                                                                                    7B
7770 1880 C5 46 00'00'21     * ****************          DO      16                                                    7B
          77 10 3E           * *                                                                                  7B
7778 1881                    * *                                                                                  7B
7778 1882 0025'2A            * *                         LHLD    TEMP_DAT        ;LOAD A DATA PATTERN INTO EXP_DAT.    7B
777B 1883 0028'22            * *                         SHLD    EXP_DAT                                               7B
777E 1884                    * *                                                                                  7B
777E 1885                    * *                                                                                  7B
777E 1886 00'28'21           * *                 4$:     LXI     H,EXP_DAT       ;WRITE A DATA PATTERN TO THE ALARM REG. 7B
7781 1887 00B5'CD            * *                         CALL    WRITE_ALARM                                           7B
7784 1888                    * *                                                                                  7B
7784 1889 00C7'CD            * *                         CALL    READ_ALARM      ;READ THE ALARM REGISTER.             7B
```

```
7787 1890                    * *
7787 1891 00'28'21           * * **************   IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11     * * *
          0798'CA 0000'CD    * * *
7795 1892                    * * *                                                ; RECEIVED DATA.
7795 1893                    * * *
7795 1894 01CE'CD            * * *   CALL    TEST21_ERROR    ;PRINT AN ERROR MESSAGE.
7798 1895                    * * *
7798 1896                    * * **************   ENDIF
7798 1897                    * *
7798 1898 0000'3A            * *     LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
779B 1899 0F                 * *     RRC
779C 1900 077E'DA            * *     JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
779F 1901                    * *
779F 1902 37                 * *     STC                     ;SET THE CARRY.
77A0 1903                    * *
77A0 1904 00'24'21           * *     LXI     H,TEMP_SHIFT    ;SHIFT THE ZERO THROUGH THE ONE FIELD.
77A3 1905 0000'22            * *     SHLD    SHIFT_PNT
77A6 1906 0000'CD            * *     CALL    SHIFTER
77A9 1907                    * *
77A9 1908 35 00'00'21        * ****************   ENDDO
          70 C1 0778'C2      *
77B2 1909                    * *
77B2 1910 35 00'00'21        ******************   ENDDO
          70 C1 06CE'C2
77BB 1911
77BB 1912 3C D3                                   TAIL
77BD 1913
77BD 1914
77BD 1915                ;*****************************************************************************
77BD 1916                ;
77BD 1917                ;
77BD 1918                ;       TEST22          DATA POINTER SEQUENCING TEST
77BD 1919                ;       ------          ----------------------------
77BD 1920                ;
77BD 1921                ;
77BD 1922                ;               THE AUTOMATIC SEQUENCING OF THE DATA POINTER IS TESTED BY
77BD 1923                ;       WRITING A DATA PATTERN TO A REGISTER IN THE ELEMENT CYCLE, SEQUENCING
77BD 1924                ;       THROUGH THE CYCLE BY READING DATA FROM THE OTHER REGISTERS IN THE
77BD 1925                ;       CYCLE, AND THEN READING THE CONTENTS OF THE FIRST REGISTER WRITTEN
77BD 1926                ;       TO. THE PROCESS IS REPEATED FOR THE HOLD CYCLE AND THE CONTROL GROUP
77BD 1927                ;       CYCLE.
77BD 1928                ;
77BD 1929                ;
77BD 1930                ;       TEST STEPS:
77BD 1931                ;
77BD 1932                ;               1) PERFORM MASTER RESET.
77BD 1933                ;               2) POINT AT CG1 MODE REGISTER.
77BD 1934                ;               3) WRITE DATA TO THIS REGISTER.
77BD 1935                ;               4) SEQUENCE THROUGH CYCLE BY DOING 14 READS.
77BD 1936                ;               5) COMPARE DATA READ FOR EACH READ WITH DATA SENT
77BD 1937                ;                  (SHOULD NOT BE EQUAL).
77BD 1938                ;               6) DO LAST READ (SHOULD BE CG1 MODE REG).
77BD 1939                ;               7) COMPARE WITH ORIGINAL DATA SENT (SHOULD BE EQUAL)
77BD 1940                ;                  AND PRINT ERRORS IF ANY.
77BD 1941                ;               8) REPEAT FOR HOLD AND CONTROL GROUP CYCLES.
77BD 1942                ;               9) DISABLE DATA POINTER SEQUENCING.
77BD 1943                ;              10) READ THEN WRITE A DATA PATTERN (SHOULD BE EQUAL).
77BD 1944                ;              11) PRINT ERRORS IF ANY.
77BD 1945
```

```
77BD 1946
77BD 1947
77BD 1948
77BD 1949
77BD 1950
77BD 1951
77BD 1952
77BD 1953
77BD 1954
77BD 1955
77BD 1956
77BD 1957
77BD 1958
77BD 1959
77BD 1960
77BD 1961
77BD 1962
77BD 1963
77BD 1964
77BD 1965
77BD 1966
77BD 1967
77BD 1968
77BD 1969
77BD 1970
77BD 1971
77BD 1972
77BD 1973
77BD 1974
77BD 1975
77BD 1976
77BD 1977
77BD 1978
77BD 1979
77BD 1980
77BD 1981
77BD 1982
77BD 1983
77BD 1984
77BD 1985
77BD 1986 0000'CD 22 3E
          0F 0000'3A
          C9 07CA'D2
          0F 0000'3A
          3D D3 09A2'DA
77D3 1987
77D3 1988 01 3E
77D5 1989 0000'32
77D8 1990
77D8 1991 0042'CD
77DB 1992
77DB 1993
77DB 1994 000E'2A
77DE 1995 0028'22
77E1 1996
77E1 1997 02 3E
77E3 1998 000B'32
77E6 1999
77E6 2000 01 3E
77E8 2001 1D D3
```

```
;       ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.


;       ERROR1: ELEMENT CYCLE SEQUENCING MODE ERROR.
;       ERROR2: HOLD CYCLE SEQUENCING MODE ERROR.
;       ERROR3: CONTROL GROUP CYCLE SEQUENCING MODE ERROR.

;       NOTE: FOR ERRORS 1-3, ADDITIONAL DATA WILL CONTAIN A NUMBER
;             TO INDICATE THE REGISTER THAT FAILED AS FOLLOWS:

;       ELEMENT CYCLE           HOLD CYCLE          CONTROL CYCLE
;       -------------           ----------          -------------

;       1 - CG1 MODE            1 - CG1 HOLD        1 - CG1 ALARM
;       2 - CG1 LOAD            2 - CG2 HOLD        2 - CG2 ALARM
;       3 - CG1 HOLD            3 - CG3 HOLD        3 - MAST MODE
;       4 - CG2 MODE            4 - CG4 HOLD
;       5 - CG2 LOAD            5 - CG5 HOLD
;       6 - CG2 HOLD
;       7 - CG3 MODE
;       8 - CG3 LOAD
;       9 - CG3 HOLD
;       A - CG4 MODE
;       B - CG4 LOAD
;       C - CG4 HOLD
;       D - CG5 MODE
;       E - CG5 LOAD
;       F - CG5 HOLD


;       ERROR4: DATA POINTER ERROR IN NON-SEQUENCING MODE.


;       DEBUG: SUSPECT INTERVAL TIMER CHIP


;*******************************************************************************

TEST22:         HEAD    <^X22>


                MVI     A,1             ;FIRST PART OF TEST 22.
                STA     CON_ERR_NUM

                CALL    MASTER_RESET_RESTORE    ;PERFORM A MASTER RESET AND
                                        ; RESTORE MASTER REG TO NORMAL VALUE

1$:             LHLD    ZERO_DAT        ;LOAD ZEROS INTO EXP_DAT.
                SHLD    EXP_DAT

                MVI     A,2             ;INDICATES SECOND ELEMENT CYCLE
                STA     CYCLE_CNT       ; REGISTER (CG1 LOAD REG).

                MVI     A,CG1_MODE_E    ;POINT TO CG1 MODE REGISTER (ELEMENT
                OUT     ITCSR           ; CYCLE).
```

```
77EA 2002
77EA 2003 0016'3A                                  LDA     TEST22_DAT      ;WRITE TWO BYTES OF ALTERNATING ONES
77ED 2004 1C D3                                    OUT     ITDB            ; AND ZEROS TO THE CG1 MODE REGISTER.
77EF 2005 1C D3                                    OUT     ITDB
77F1 2006
77F1 2007 C5 46 00'00'21      ******************   DO      14
          77 0E 3E            *
77F9 2008                     *
77F9 2009 1C DB               *                    IN      ITDB            ;READ LOW BYTE AN ELEMENT CYCLE
77FB 2010 002C'32             *                    STA     REC_DAT+1       ; REGISTER (NOT CG1 MODE REG).
77FE 2011                     *
77FE 2012 1C DB               *                    IN      ITDB            ;READ HIGH BYTE OF SAME REGISTER.
7800 2013 F4 E6               *                    ANI     <^XF4>          ;CLEAR BITS 8,9 AND 11 IF IT IS
7802 2014 002B'32             *                    STA     REC_DAT         ; A MODE REG.
7805 2015                     *
7805 2016 00'28'21            * ****************   IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11      * *
          0816'CA 0000'CD     * *
7813 2017                     * *                                                   ; RECEIVED DATA.
7813 2018                     * *
7813 2019 020D'CD             * *                  CALL    TEST22_ERROR    ;PRINT AN ERROR MESSAGE.
7816 2020                     * *
7816 2021                     * ****************   ENDIF
7816 2022                     *
7816 2023 0100'CD             *                    CALL    INC_CYCLE_CNT   ;INCREMENT THE CYCLE COUNTER.
7819 2024                     *
7819 2025 35 00'00'21         ******************   ENDDO
          70 C1 07F9'C2
7822 2026
7822 2027 000B'32 3C AF                            SET     CYCLE_CNT       ;INDICATES FIRST ELEMENT CYCLE
7827 2028                                                                  ; REGISTER (CG1 MODE REG).
7827 2029
7827 2030 0016'2A                                  LHLD    TEST22_DAT      ;LOAD PATTERN OF ONES AND ZEROS
782A 2031 0028'22                                  SHLD    EXP_DAT         ; INTO EXP_DAT.
782D 2032
782D 2033 1C DB                                    IN      ITDB            ;READ LOW BYTE OF WHAT SHOULD BE
782F 2034 002C'32                                  STA     REC_DAT+1       ; CG1 MODE REGISTER.
7832 2035
7832 2036 1C DB                                    IN      ITDB            ;READ HIGH BYTE OF SAME REGISTER.
7834 2037 002B'32                                  STA     REC_DAT
7837 2038
7837 2039 00'28'21            ******************   IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11      *
          0848'CA 0000'CD     *
7845 2040                     *                                                   ; RECEIVED DATA.
7845 2041                     *
7845 2042 020D'CD             *                    CALL    TEST22_ERROR    ;PRINT AN ERROR MESSAGE.
7848 2043                     *
7848 2044                     ******************   ENDIF
7848 2045
7848 2046 0000'3A                                  LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
784B 2047 0F                                       RRC
784C 2048 07DB'DA                                  JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
784F 2049
784F 2050
784F 2051 02 3E                                    MVI     A,2             ;SECOND PART OF TEST 22.
7851 2052 0000'32                                  STA     CON_ERR_NUM
7854 2053
7854 2054 002D'CD                                  CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7857 2055
```

```
7857 2056 0042'CD                                  CALL    MASTER_RESET_RESTORE    ;PERFORM A MASTER RESET AND
785A 2057                                                                          ; RESTORE MASTER REG TO NORMAL VALUE
785A 2058
785A 2059 000E'2A                          2$:     LHLD    ZERO_DAT                ;LOAD ZEROS INTO EXP_DAT.
785D 2060 0028'22                                  SHLD    EXP_DAT
7860 2061
7860 2062 02 3E                                    MVI     A,2                     ;INDICATES SECOND REGISTER IN
7862 2063 000B'32                                  STA     CYCLE_CNT               ; HOLD CYCLE (CG2 HOLD REG).
7865 2064
7865 2065 19 3E                                    MVI     A,CG1_HOLD_H            ;POINT TO CG1 HOLD REGISTER (HOLD
7867 2066 1D D3                                    OUT     ITCSR                   ; CYCLE).
7869 2067
7869 2068 0016'3A                                  LDA     TEST22_DAT              ;WRITE TWO BYTES OF ALTERNATING ONES
786C 2069 1C D3                                    OUT     ITDB                    ; AND ZEROS TO THE CG1 HOLD REGISTER.
786E 2070 1C D3                                    OUT     ITDB
7870 2071
7870 2072 C5 46 00'00'21          ******************       DO      4
          77 04 3E                *
7878 2073                         *
7878 2074 1C DB                   *                        IN      ITDB                    ;READ LOW BYTE AN ELEMENT CYCLE
787A 2075 002C'32                 *                        STA     REC_DAT+1               ; REGISTER (NOT CG1 HOLD REG).
787D 2076                         *
787D 2077 1C DB                   *                        IN      ITDB                    ;READ HIGH BYTE OF SAME REGISTER.
787F 2078 002B'32                 *                        STA     REC_DAT
7882 2079                         *
7882 2080 00'28'21                * ****************       IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11          * *
          0893'CA 0000'CD         * *
7890 2081                         * *                                                      ; RECEIVED DATA.
7890 2082                         * *
7890 2083 020D'CD                 * *                        CALL    TEST22_ERROR            ;PRINT AN ERROR MESSAGE.
7893 2084                         * *
7893 2085                         * ****************       ENDIF
7893 2086                         *
7893 2087 0100'CD                 *                        CALL    INC_CYCLE_CNT           ;INCREMENT THE CYCLE COUNTER.
7896 2088                         *
7896 2089 35 00'00'21             ******************       ENDDO
          70 C1 0878'C2
789F 2090
789F 2091 000B'32 3C AF                                    SET     CYCLE_CNT               ;INDICATES FIRST HOLD CYCLE
78A4 2092                                                                                  ; REGISTER (CG1 HOLD REG).
78A4 2093
78A4 2094 0016'2A                                          LHLD    TEST22_DAT              ;LOAD PATTERN OF ONES AND ZEROS
78A7 2095 0028'22                                          SHLD    EXP_DAT                 ; INTO EXP_DAT.
78AA 2096
78AA 2097 1C DB                                            IN      ITDB                    ;READ LOW BYTE OF WHAT SHOULD BE
78AC 2098 002C'32                                          STA     REC_DAT+1               ; CG1 HOLD REGISTER.
78AF 2099
78AF 2100 1C DB                                            IN      ITDB                    ;READ HIGH BYTE OF SAME REGISTER.
78B1 2101 002B'32                                          STA     REC_DAT
78B4 2102
78B4 2103 00'28'21                ******************       IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11          *
          08C5'CA 0000'CD         *
78C2 2104                         *                                                      ; RECEIVED DATA.
78C2 2105                         *
78C2 2106 020D'CD                 *                        CALL    TEST22_ERROR            ;PRINT AN ERROR MESSAGE.
78C5 2107                         *
78C5 2108                         ******************       ENDIF
78C5 2109
```

```
78C5 2110 0000'3A                                  LDA     ERROR_CON          ;IF THE ERROR LOOPING FLAG IS SET...
78C8 2111 0F                                       RRC
78C9 2112 085A'DA                                  JC      2$                 ;...JUMP TO BEGINNING OF ERROR LOOP.
78CC 2113
78CC 2114
78CC 2115
78CC 2116 03 3E                                    MVI     A,3                ;THIRD PART OF TEST 22.
78CE 2117 0000'32                                  STA     CON_ERR_NUM
78D1 2118
78D1 2119 0C2D'CD                                  CALL    CHECK_RD           ;CALL BOOT CODE IF UNDER RD CONTROL
78D4 2120
78D4 2121 0042'CD                                  CALL    MASTER_RESET_RESTORE     ;PERFORM A MASTER RESET AND
78D7 2122                                                                     ; RESTORE MASTER REG TO NORMAL VALUE
78D7 2123
78D7 2124 000E'2A                        3$:       LHLD    ZERO_DAT           ;LOAD ZEROS INTO EXP_DAT.
78DA 2125 0028'22                                  SHLD    EXP_DAT
78DD 2126
78DD 2127 02 3E                                    MVI     A,2                ;INDICATES SECOND CONTROL GROUP
78DF 2128 000B'32                                  STA     CYCLE_CNT          ; CYCLE REGISTER (CG2 ALARM REG).
78E2 2129
78E2 2130 07 3E                                    MVI     A,ALARM_REG1       ;POINT TO CG1 ALARM REGISTER (CONTROL
78E4 2131 1D D3                                    OUT     ITCSR              ; GROUP CYCLE).
78E6 2132
78E6 2133 0016'3A                                  LDA     TEST22_DAT         ;WRITE TWO BYTES OF ALTERNATING ONES
78E9 2134 1C D3                                    OUT     ITDB               ; AND ZEROS TO THE CG1 HOLD REGISTER.
78EB 2135 1C D3                                    OUT     ITDB
78ED 2136
78ED 2137 1C DB                                    IN      ITDB               ;READ LOW BYTE A CONTROL GROUP
78EF 2138 002C'32                                  STA     REC_DAT+1          ; REGISTER (NOT ALARM 1 REG).
78F2 2139
78F2 2140 1C DB                                    IN      ITDB               ;READ HIGH BYTE OF SAME REGISTER.
78F4 2141 002B'32                                  STA     REC_DAT
78F7 2142
78F7 2143 00'28'21                 ******************IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11           *
          0908'CA 0000'CD          *
7905 2144                          *                                          ; RECEIVED DATA.
7905 2145                          *
7905 2146 020D'CD                  *               CALL    TEST22_ERROR       ;PRINT AN ERROR MESSAGE.
7908 2147                          *
7908 2148                          ******************ENDIF
7908 2149
7908 2150 0100'CD                                  CALL    INC_CYCLE_CNT      ;INCREMENT THE CYCLE COUNTER.
7908 2151
7908 2152
7908 2153 000C'2A                                  LHLD    NORMAL_MASTER      ;GET EXPECTED DATA (MASTER MODE DATA)
790E 2154 0028'22                                  SHLD    EXP_DAT            ;PUT IN EXPECTED DATA
7911 2155
7911 2156 1C DB                                    IN      ITDB               ;READ LOW BYTE A CONTROL GROUP
7913 2157 002C'32                                  STA     REC_DAT+1          ; REGISTER (NOT ALARM 1 REG).
7916 2158
7916 2159 1C DB                                    IN      ITDB               ;READ HIGH BYTE OF SAME REGISTER.
7918 2160 002B'32                                  STA     REC_DAT
791B 2161
791B 2162 00'28'21                 ******************IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11           *
          092C'CA 0000'CD          *
7929 2163                          *                                          ; RECEIVED DATA.
7929 2164                          *
7929 2165 020D'CD                  *               CALL    TEST22_ERROR       ;PRINT AN ERROR MESSAGE.
```

```
792C 2166                    *
792C 2167                    ******************          ENDIF
792C 2168
792C 2169
792C 2170 000B'32 3C AF                                  SET     CYCLE_CNT       ;INDICATES FIRST CONTROL GROUP
7931 2171                                                                        ; CYCLE REGISTER (CG1 ALARM REG).
7931 2172
7931 2173 0016'2A                                        LHLD    TEST22_DAT      ;LOAD PATTERN OF ONES AND ZEROS
7934 2174 0028'22                                        SHLD    EXP_DAT         ; INTO EXP_DAT.
7937 2175
7937 2176 1C DB                                          IN      ITDB            ;READ LOW BYTE OF WHAT SHOULD BE
7939 2177 002C'32                                        STA     REC_DAT+1       ; CG1 ALARM REGISTER.
793C 2178
793C 2179 1C DB                                          IN      ITDB            ;READ HIGH BYTE OF SAME REGISTER.
793E 2180 002B'32                                        STA     REC_DAT
7941 2181
7941 2182 00'28'21           ******************          IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11     *
          0952'CA 0000'CD    *
794F 2183                    *                                                           ; RECEIVED DATA.
794F 2184                    *
794F 2185 020D'CD            *                           CALL    TEST22_ERROR    ;PRINT AN ERROR MESSAGE.
7952 2186                    *
7952 2187                    ******************          ENDIF
7952 2188
7952 2189 0000'3A                                        LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7955 2190 0F                                             RRC
7956 2191 08D7'DA                                        JC      3$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7959 2192
7959 2193
7959 2194 04 3E                                          MVI     A,4             ;FOURTH PART OF TEST 22.
795B 2195 0000'32                                        STA     CON_ERR_NUM
795E 2196
795E 2197 002D'CD                                        CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7961 2198
7961 2199 0042'CD                                        CALL    MASTER_RESET_RESTORE    ;PERFORM A MASTER RESET AND
7964 2200                                                                        ; RESTORE MASTER REG TO NORMAL VALUE
7964 2201
7964 2202 0016'2A                                        LHLD    TEST22_DAT      ;LOAD PATTERN OF ONES AND ZEROS
7967 2203 0028'22                                        SHLD    EXP_DAT         ; INTO EXP_DAT
796A 2204
796A 2205 01 3E                              4$:         MVI     A,CG1_MODE_E    ;POINT TO CG1 MODE REGISTER (ELEMENT
796C 2206 1D D3                                          OUT     ITCSR           ; CYCLE).
796E 2207
796E 2208 E8 3E                                          MVI     A,DIS_DP_SEQ    ;DISABLE DATA POINTER SEQUENCING.
7970 2209 1D D3                                          OUT     ITCSR
7972 2210
7972 2211 0016'3A                                        LDA     TEST22_DAT      ;WRITE TWO BYTES OF ALTERNATING ONES
7975 2212 1C D3                                          OUT     ITDB            ; AND ZEROS TO THE CG1 MODE REGISTER.
7977 2213 1C D3                                          OUT     ITDB
7979 2214
7979 2215 1C DB                                          IN      ITDB            ;READ THE LOW BYTE OF THE CG1 MODE
797B 2216 002C'32                                        STA     REC_DAT+1       ; REGISTER.
797E 2217
797E 2218 1C DB                                          IN      ITDB            ;AND THE HIGH BYTE
7980 2219 002B'32                                        STA     REC_DAT
7983 2220
7983 2221 00'28'21           ******************          IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11     *
          0999'CA 0000'CD    *
```

```
7991 2222                    *                                  ; RECEIVED DATA.
7991 2223                    *
7991 2224 0000'32 3C AF      *                    SET     NA_OTHER        ;NO ADDITIONAL DATA.
7996 2225                    *
7996 2226 020D'CD            *                    CALL    TEST22_ERROR    ;PRINT AN ERROR MESSAGE.
7999 2227                    *
7999 2228                    ******************   ENDIF
7999 2229
7999 2230 0000'3A                                 LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
799C 2231 0F                                      RRC
799D 2232 096A'DA                                 JC      4$              ;...JUMP TO BEGINNING OF ERROR LOOP.
79A0 2233
79A0 2234
79A0 2235 3C D3                                   TAIL
79A2 2236
79A2 2237
79A2 2238                    ;*****************************************************************************
79A2 2239                    ;
79A2 2240                    ;
79A2 2241                    ;       TEST23          COUNTER LOAD AND SAVE TEST
79A2 2242                    ;       ------          --------------------------
79A2 2243                    ;
79A2 2244                    ;
79A2 2245                    ;               THIS TEST LOADS EACH OF THE LOAD REGISTERS WITH DATA PATTERNS
79A2 2246                    ;       AND THEN LOADS ALL THE COUNTERS FROM THE LOAD REGISTERS. THE CONTENTS
79A2 2247                    ;       OF THE COUNTERS ARE THEN SAVED IN THE HOLD REGISTERS, AND READ BACK.
79A2 2248                    ;       THE RECEIVED DATA IS COMPARED WITH THE DATA SENT AND ANY APPROPRIATE
79A2 2249                    ;       ERROR MESSAGES ARE PRINTED. THE DATA PATTERNS CONSIST OF ALL ZEROS
79A2 2250                    ;       ALL ONES, AND A ONE SHIFTED THROUGH A ZERO FIELD.
79A2 2251                    ;
79A2 2252                    ;
79A2 2253                    ;       TEST STEPS:
79A2 2254                    ;
79A2 2255                    ;               1) PERFORM MASTER RESET.
79A2 2256                    ;               2) POINT TO THE CG1 LOAD REGISTER.
79A2 2257                    ;               3) WRITE 16-BIT DATA PATTERN TO LOAD REGISTER.
79A2 2258                    ;               5) LOAD COUNTER 1 FROM LOAD REGISTER 1.
79A2 2259                    ;               6) SAVE COUNTER 1 IN HOLD REGISTER.
79A2 2260                    ;               7) POINT TO CG1 HOLD REGISTER.
79A2 2261                    ;               8) READ HOLD REGISTER 1 AND COMPARE WITH DATA SENT.
79A2 2262                    ;               9) PRINT ERROR OF ANY.
79A2 2263                    ;               9) REPEAT FOR EVERY DATA PATTERN.
79A2 2264                    ;              10) REPEAT FOR COUNTER GROUPS 2,3,4 AND 5.
79A2 2265                    ;
79A2 2266                    ;
79A2 2267                    ;       ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.
79A2 2268                    ;
79A2 2269                    ;
79A2 2270                    ;       ERROR1: LOAD/SAVE TEST ERROR. ADDITIONAL DATA WILL SPECIFY COUNTER
79A2 2271                    ;               GROUP.
79A2 2272                    ;
79A2 2273                    ;
79A2 2274                    ;       DEBUG: SUSPECT INTERVAL TIMER CHIP.
79A2 2275                    ;
79A2 2276                    ;
79A2 2277                    ;*****************************************************************************
79A2 2278
79A2 2279
79A2 2280 0000'CD 23 3E      TEST23:         HEAD    <^X23>
          0F 0000'3A
```

```
          C9 09AF'D2
          0F 0000'3A
          3D D3 0AAE'DA
79B8 2281
79B8 2282 003D'CD                                  CALL    MASTER_RESET     ;PERFORM A MASTER RESET.
79BB 2283
79BB 2284 00'18'21                                 LXI     H,TEST23_DAT     ;INITIALIZE THE MASTER MODE
79BE 2285 0057'CD                                  CALL    WRITE_MASTER     ; REGISTER TO 0A00
79C1 2286
79C1 2287 004D'CD                                  CALL    CLR_CG_PNT       ;CLEAR THE COUNTER GROUP POINTER.
79C4 2288
79C4 2289 C5 46 00'00'21      ******************   DO      5
          77 05 3E            *
79CC 2290                     *
79CC 2291 0052'CD             *                    CALL    INC_CG_PNT       ;INCREMENT THE COUNTER GROUP POINTER.
79CF 2292                     *
79CF 2293 00'1A'21            *                    LXI     H,TEST23_DAT+2   ;INITIALIZE EACH COUNTER MODE REG.
79D2 2294 006D'CD             *                    CALL    WRITE_MODE
79D5 2295                     *
79D5 2296 35 00'00'21         ******************   ENDDO
          70 C1 09CC'C2
79DE 2297
79DE 2298 004D'CD                                  CALL    CLR_CG_PNT       ;CLEAR THE COUNTER GROUP POINTER.
79E1 2299
79E1 2300 C5 46 00'00'21      ******************   DO      5
          77 05 3E            *
79E9 2301                     *
79E9 2302 0052'CD             *                    CALL    INC_CG_PNT       ;INCREMENT THE COUNTER GROUP POINTER.
79EC 2303                     *
79EC 2304 000E'2A             *                    LHLD    ZERO_DAT         ;MOVE ALL ZEROS TO EXP_DAT.
79EF 2305 0028'22             *                    SHLD    EXP_DAT
79F2 2306                     *
79F2 2307 00'28'21            *            1$:     LXI     H,EXP_DAT        ;WRITE ALL ZEROS TO THE LOAD REG.
79F5 2308 0085'CD             *                    CALL    WRITE_LOAD
79F8 2309                     *
79F8 2310 0105'CD             *                    CALL    DISARM_CNTR      ;DISARM PRESENT COUNTER.
79FB 2311                     *
79FB 2312 011A'CD             *                    CALL    LOAD_CNTR        ;LOAD PRESENT COUNTER FROM LOAD REG.
79FE 2313                     *
79FE 2314 012F'CD             *                    CALL    SAVE_CNTR        ;SAVE PRESENT COUNTER IN HOLD REGISTER.
7A01 2315                     *
7A01 2316 00A9'CD             *                    CALL    READ_HOLD        ;READ THE HOLD REGISTER.
7A04 2317                     *
7A04 2318 00'28'21            * ****************   IFNEQ   EXP_DAT,REC_DAT,2        ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11      * *
          0A15'CA 0000'CD     * *
7A12 2319                     * *                                                  ; RECEIVED DATA.
7A12 2320                     * *
7A12 2321 01CE'CD             * *                  CALL    TEST21_ERROR     ;PRINT AN ERROR MESSAGE.
7A15 2322                     * *
7A15 2323                     * ****************   ENDIF
7A15 2324                     *
7A15 2325 0000'3A             *                    LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
7A18 2326 0F                  *                    RRC
7A19 2327 09F2'DA             *                    JC      1$               ;...JUMP TO BEGINNING OF ERROR LOOP.
7A1C 2328                     *
7A1C 2329                     *
7A1C 2330 0011'2A             *                    LHLD    ONE_DAT          ;MOVE ALL ONES TO EXP_DAT.
7A1F 2331 0028'22             *                    SHLD    EXP_DAT
7A22 2332                     *
```

```
7A22 2333 00'28'21          *                2$:     LXI     H,EXP_DAT       ;WRITE ALL ONES TO THE TIMER.
7A25 2334 0085'CD           *                        CALL    WRITE_LOAD
7A28 2335                   *
7A28 2336 0105'CD           *                        CALL    DISARM_CNTR     ;DISARM PRESENT COUNTER.
7A2B 2337                   *
7A2B 2338 011A'CD           *                        CALL    LOAD_CNTR       ;LOAD PRESENT COUNTER FROM LOAD REG.
7A2E 2339                   *
7A2E 2340 012F'CD           *                        CALL    SAVE_CNTR       ;SAVE PRESENT COUNTER IN HOLD REGISTER.
7A31 2341                   *
7A31 2342 00A9'CD           *                        CALL    READ_HOLD       ;READ THE HOLD REGISTER.
7A34 2343                   *
7A34 2344 00'28'21          * ****************       IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11    * *
          0A45'CA 0000'CD   * *
7A42 2345                   * *                                                      ; RECEIVED DATA.
7A42 2346                   * *
7A42 2347 01CE'CD           * *                      CALL    TEST21_ERROR    ;PRINT AN ERROR MESSAGE.
7A45 2348                   * *
7A45 2349                   * ****************       ENDIF
7A45 2350                   *
7A45 2351 0000'3A           *                        LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7A48 2352 0F                *                        RRC
7A49 2353 0A22'DA           *                        JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7A4C 2354                   *
7A4C 2355 000F'2A           *                        LHLD    ONE_SHIFT       ;LOAD A ONE SHIFTED THROUGH ZERO
7A4F 2356 0025'22           *                        SHLD    TEMP_DAT        ; STRING INTO TEMP_DAT.
7A52 2357                   *
7A52 2358 C5 46 00'00'21    * ****************       DO      16
          77 10 3E          * *
7A5A 2359                   * *
7A5A 2360 002D'CD           * *                      CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7A5D 2361                   * *
7A5D 2362 0025'2A           * *                      LHLD    TEMP_DAT        ;LOAD A DATA PATTERN INTO EXP_DAT.
7A60 2363 0028'22           * *                      SHLD    EXP_DAT
7A63 2364                   * *
7A63 2365 00'28'21          * *                3$:   LXI     H,EXP_DAT       ;WRITE A DATA PATTERN TO THE TIMER.
7A66 2366 0085'CD           * *                      CALL    WRITE_LOAD
7A69 2367                   * *
7A69 2368 0105'CD           * *                      CALL    DISARM_CNTR     ;DISARM PRESENT COUNTER.
7A6C 2369                   * *
7A6C 2370 011A'CD           * *                      CALL    LOAD_CNTR       ;LOAD PRESENT COUNTER FROM LOAD REG.
7A6F 2371                   * *
7A6F 2372 0105'CD           * *                      CALL    DISARM_CNTR     ;DISARM PRESENT COUNTER.
7A72 2373                   * *
7A72 2374 012F'CD           * *                      CALL    SAVE_CNTR       ;SAVE PRESENT COUNTER IN HOLD REGISTER.
7A75 2375                   * *
7A75 2376 00A9'CD           * *                      CALL    READ_HOLD       ;READ THE HOLD REGISTER.
7A78 2377                   * *
7A78 2378 00'28'21          * * **************       IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11    * * *
          0A89'CA 0000'CD   * * *
7A86 2379                   * * *                                                    ; RECEIVED DATA.
7A86 2380                   * * *
7A86 2381 01CE'CD           * * *                    CALL    TEST21_ERROR    ;PRINT AN ERROR MESSAGE.
7A89 2382                   * * *
7A89 2383                   * * **************       ENDIF
7A89 2384                   * *
7A89 2385 0000'3A           * *                      LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7A8C 2386 0F                * *                      RRC
7A8D 2387 0A63'DA           * *                      JC      3$              ;...JUMP TO BEGINNING OF ERROR LOOP.
```

```
7A90 2388                    * *
7A90 2389 B7                 * *                    ORA     A               ;CLEAR THE CARRY.
7A91 2390                    * *
7A91 2391 00'24'21           * *                    LXI     H,TEMP_SHIFT    ;SHIFT THE ONE THROUGH THE ZERO FIELD.
7A94 2392 0000'22            * *                    SHLD    SHIFT_PNT
7A97 2393 0000'CD            * *                    CALL    SHIFTER
7A9A 2394                    * *
7A9A 2395 35 00'00'21        * ****************     ENDDO
          70 C1 0A5A'C2      *
7AA3 2396                    *
7AA3 2397 35 00'00'21        ******************     ENDDO
          70 C1 09E9'C2
7AAC 2398
7AAC 2399 3C D3                                     TAIL
7AAE 2400
7AAE 2401
7AAE 2402
7AAE 2403                    ;*****************************************************************************
7AAE 2404                    ;
7AAE 2405                    ;
7AAE 2406                    ;
7AAE 2407                    ;       TEST24                  OUTPUT BIT SET/CLEAR TEST
7AAE 2408                    ;       ------                  -------------------------
7AAE 2409                    ;
7AAE 2410                    ;               THIS TEST ENSURES THAT THE COUNTER OUTPUTS CAN BE SET AND
7AAE 2411                    ;       CLEARED BY USING THE COMMAND REGISTER COMMAND SET OUTPUT AND CLEAR
7AAE 2412                    ;       OUTPUT. THE COUNTER OUTPUTS ARE READ THROUGH THE STATUS REGISTER.
7AAE 2413                    ;       THIS TEST IS PERFORMED IN THE ACTIVE HIGH DELAYED TOGGLE MODE.
7AAE 2414                    ;
7AAE 2415                    ;
7AAE 2416                    ;       TEST STEPS:
7AAE 2417                    ;
7AAE 2418                    ;               1) PERFORM MASTER RESET.
7AAE 2419                    ;               2) SET CG MODE REGISTERS TO ACTIVE HIGH DELAYED TOGGLE.
7AAE 2420                    ;               3) PERFORM SET OUTPUT BIT COMMAND FOR EACH COUNTER GROUP.
7AAE 2421                    ;               4) READ STATUS REGISTER (SHOULD BE HEX FF).
7AAE 2422                    ;               5) PRINT ERRORS IF ANY.
7AAE 2423                    ;               6) PERFORM CLEAR OUTPUT BIT COMMAND FOR EACH COUNTER GROUP.
7AAE 2424                    ;               7) READ STATUS REGISTER (SHOULD BE HEX C1).
7AAE 2425                    ;               8) PRINT ERRORS IF ANY.
7AAE 2426                    ;
7AAE 2427                    ;
7AAE 2428                    ;       ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.
7AAE 2429                    ;
7AAE 2430                    ;
7AAE 2431                    ;       ERROR1: COUNTER OUTPUT FAILURE, OUTPUT(S) NOT BEING SET.
7AAE 2432                    ;       ERROR2: COUNTER OUTPUT FAILURE, OUTPUT(S) NOT BEING CLEARED.
7AAE 2433                    ;
7AAE 2434                    ;
7AAE 2435                    ;       DEBUG: SUSPECT INTERVAL TIMER CHIP.
7AAE 2436                    ;
7AAE 2437                    ;
7AAE 2438                    ;*****************************************************************************
7AAE 2439
7AAE 2440
7AAE 2441 0000'CD 24 3E      TEST24:         HEAD    <^X24>
          0F 0000'3A
          C9 0ABB'D2
          0F 0000'3A
          3D D3 0B77'DA
```

```
7AC4 2442
7AC4 2443 01 3E                                  MVI     A,1                 ;FIRST PART OF TEST 8.
7AC6 2444 0000'32                                STA     CON_ERR_NUM
7AC9 2445
7AC9 2446 0042'CD                                CALL    MASTER_RESET_RESTORE    ;PERFORM A MASTER RESET AND
7ACC 2447                                                                        ; RESTORE MASTER REG TO NORMAL VALUE
7ACC 2448
7ACC 2449 004D'CD                                CALL    CLR_CG_PNT          ;CLEAR THE COUNTER GROUP POINTER.
7ACF 2450
7ACF 2451 C5 46 00'00'21      ******************  DO      5
          77 05 3E            *
7AD7 2452                     *
7AD7 2453 0052'CD             *                   CALL    INC_CG_PNT          ;INCREMENT THE COUNTER GROUP POINTER.
7ADA 2454                     *
7ADA 2455 00'1C'21            *                   LXI     H,TEST24_DAT        ;SET EACH COUNTER OUTPUT TO DELAYED
7ADD 2456 006D'CD             *                   CALL    WRITE_MODE          ; TOGGLE.
7AE0 2457                     *
7AE0 2458 35 00'00'21         ******************  ENDDO
          70 C1 0AD7'C2
7AE9 2459
7AE9 2460 3E 3E                                  MVI     A,<^X3E>            ;LOAD DATA WITH ALL OUTPUT BITS
7AEB 2461 0028'32                                STA     EXP_DAT             ; SET INTO EXP_DAT.
7AEE 2462
7AEE 2463 004D'CD                          1$:   CALL    CLR_CG_PNT          ;CLEAR THE COUNTER GROUP POINTER.
7AF1 2464
7AF1 2465 C5 46 00'00'21      ******************  DO      5
          77 05 3E            *
7AF9 2466                     *
7AF9 2467 0052'CD             *                   CALL    INC_CG_PNT          ;INCREMENT THE COUNTER GROUP POINTER.
7AFC 2468                     *
7AFC 2469 0105'CD             *                   CALL    DISARM_CNTR         ;DISARM THE COUNTER.
7AFF 2470                     *
7AFF 2471 0144'CD             *                   CALL    SET_OUTPUT          ;SET THE COUNTER'S OUTPUT.
7B02 2472                     *
7B02 2473 35 00'00'21         ******************  ENDDO
          70 C1 0AF9'C2
7B0B 2474
7B0B 2475 1D DB                                  IN      ITCSR               ;READ THE STATUS REGISTER.
7B0D 2476 3E E6                                  ANI     <^X3E>              ;MASK OUT BITS 15,14 AND 0.
7B0F 2477 002B'32                                STA     REC_DAT             ;STORE IN  REC_DAT.
7B12 2478
7B12 2479 00'28'21            ******************  IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'2B'11      *
          0B23'CA 0000'CD     *
7B20 2480                     *                                                  ; RECEIVED DATA.
7B20 2481                     *
7B20 2482 0247'CD             *                   CALL    TEST24_ERROR        ;PRINT AN ERROR MESSAGE.
7B23 2483                     *
7B23 2484                     ******************  ENDIF
7B23 2485
7B23 2486 0000'3A                                LDA     ERROR_CON           ;IF THE ERROR LOOPING FLAG IS SET...
7B26 2487 0F                                     RRC
7B27 2488 0AEE'DA                                JC      1$                  ;...JUMP TO BEGINNING OF ERROR LOOP.
7B2A 2489
7B2A 2490
7B2A 2491 002D'CD                                CALL    CHECK_RD            ;CALL BOOT CODE IF UNDER RD CONTROL
7B2D 2492
7B2D 2493 02 3E                                  MVI     A,2                 ;SECOND PART OF TEST 8.
7B2F 2494 0000'32                                STA     CON_ERR_NUM
7B32 2495
```

```
7B32 2496 0028'32 AF                                          CLEAR   EXP_DAT          ;ALL OUTPUT BITS CLEAR.
7B36 2497
7B36 2498 004D'CD                                2$:          CALL    CLR_CG_PNT       ;CLEAR THE COUNTER GROUP POINTER.
7B39 2499
7B39 2500 C5 46 00'00'21      ******************              DO      5
          77 05 3E            *
7B41 2501                     *
7B41 2502 0052'CD             *                                CALL    INC_CG_PNT       ;INCREMENT THE COUNTER GROUP POINTER.
7B44 2503                     *
7B44 2504 0105'CD             *                                CALL    DISARM_CNTR      ;DISARM THE COUNTER.
7B47 2505                     *
7B47 2506 0144'CD             *                                CALL    SET_OUTPUT       ;SET THE COUNTER'S OUTPUT.
7B4A 2507                     *
7B4A 2508 014D'CD             *                                CALL    CLR_OUTPUT       ;CLEAR THE COUNTER'S OUTPUT.
7B4D 2509                     *
7B4D 2510 35 00'00'21         ******************              ENDDO
          70 C1 0B41'C2
7B56 2511
7B56 2512 1D DB                                                IN      ITCSR            ;READ THE STATUS REGISTER (ALL OUTPUT
7B58 2513 3E E6                                                ANI     <^X3E>           ;MASK OUT BITS 15,14 AND 0.
7B5A 2514 002B'32                                              STA     REC_DAT          ; BITS SHOULD BE CLEAR).
7B5D 2515
7B5D 2516 00'28'21            ******************              IFNEQ   EXP_DAT,REC_DAT,1        ;COMPARE THE EXPECTED AND
          01 0E 00'2B'11      *
          0B6E'CA 0000'CD     *
7B6B 2517                     *                                                                 ; RECEIVED DATA.
7B6B 2518                     *
7B6B 2519 0247'CD             *                                CALL    TEST24_ERROR     ;PRINT AN ERROR MESSAGE.
7B6E 2520                     *
7B6E 2521                     ******************              ENDIF
7B6E 2522
7B6E 2523 0000'3A                                              LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
7B71 2524 0F                                                   RRC
7B72 2525 0B36'DA                                              JC      2$               ;...JUMP TO BEGINNING OF ERROR LOOP.
7B75 2526
7B75 2527
7B75 2528 3C D3                                                TAIL
7B77 2529
7B77 2530
7B77 2531
7B77 2532                                ;*****************************************************************************
7B77 2533                                ;
7B77 2534                                ;
7B77 2535                                ;       TEST25                  COUNTER CARRY TEST
7B77 2536                                ;       ------                  ------------------
7B77 2537                                ;
7B77 2538                                ;
7B77 2539                                ;               THIS TEST LOADS DATA PATTERNS INTO THE COUNTER REGISTERS
7B77 2540                                ;       THAT WILL CAUSE A CARRY TO THE NEXT HIGHER BIT. THE COUNTER IS THEN
7B77 2541                                ;       INCREMENTED, AND THE RESULT CHECKED TO DETERMINE IF THE COUNTER IS IN
7B77 2542                                ;       FACT COUNTING PROPERLY.
7B77 2543                                ;
7B77 2544                                ;
7B77 2545                                ;       DATA LOADED: 0000000000000000     DATA EXPECTED: 0000000000000001
7B77 2546                                ;                    0000000000000001                    0000000000000010
7B77 2547                                ;                    0000000000000011                    0000000000000100
7B77 2548                                ;                    0000000000000111                    0000000000001000
7B77 2549                                ;                    0000000000001111                    0000000000010000
7B77 2550                                ;                    0000000000011111                    0000000000100000
7B77 2551                                ;                    0000000000111111                    0000000001000000
```

```
7B77 2552
7B77 2553
7B77 2554
7B77 2555
7B77 2556
7B77 2557
7B77 2558
7B77 2559
7B77 2560
7B77 2561
7B77 2562
7B77 2563
7B77 2564
7B77 2565
7B77 2566
7B77 2567
7B77 2568
7B77 2569
7B77 2570
7B77 2571
7B77 2572
7B77 2573
7B77 2574
7B77 2575
7B77 2576
7B77 2577
7B77 2578
7B77 2579
7B77 2580
7B77 2581
7B77 2582
7B77 2583
7B77 2584
7B77 2585
7B77 2586
7B77 2587
7B77 2588
7B77 2589
7B77 2590
7B77 2591 0000'CD 25 3E
          0F 0000'3A
          C9 0884'D2
          0F 0000'3A
          3D D3 0C23'DA
7B8D 2592
7B8D 2593 0042'CD
7B90 2594
7B90 2595
7B90 2596 004D'CD
7B93 2597
7B93 2598 C5 46 00'00'21
          77 05 3E
7B9B 2599
7B9B 2600 0052'CD
7B9E 2601
7B9E 2602 00'1E'21
7BA1 2603 006D'CD
7BA4 2604
7BA4 2605 35 00'00'21
          70 C1 0B9B'C2
```

```
                    0000000001111111                    0000000010000000
                    0000000011111111                    0000000100000000
                    0000000111111111                    0000001000000000
                    0000001111111111                    0000010000000000
                    0000011111111111                    0000100000000000
                    0000111111111111                    0001000000000000
                    0001111111111111                    0010000000000000
                    0011111111111111                    0100000000000000
                    0111111111111111                    1000000000000000


TEST STEPS:

        1) PERFORM MASTER RESET.
        2) WRITE HEX 0010 TO ALL COUNTER MODE REGISTER TO COUNT UP.
        3) WRITE A DATA PATTERN TO COUNTER LOAD REGISTER.
        4) LOAD COUNTER FROM ASSOCIATED LOAD REGISTER.
        5) INCREMENT COUNTER USING COMMAND REGISTER COUNTER STEP.
        6) SAVE COUNTER'S CONTENTS IN ASSOCIATED HOLD REGISTER.
        7) READ THE HOLD REGISTER AND COMPARE DATA WITH EXPECTED DATA.
        8) PRINT ERRORS IF ANY.
        9) REPEAT STEPS 3 - 8 FOR EACH DATA PATTERN.
       10) REPEAT 1-9 FOR EACH COUNTER GROUP.


ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.


ERROR1: COUNTER INCREMENT TEST FAILURE. ADDITIONAL DATA WILL SPECIFY
        COUNTER GROUP THAT FAILED.


DEBUG: SUSPECT INTERVAL TIMER CHIP.

;*****************************************************************************

TEST25:         HEAD    <^X25>


                CALL    MASTER_RESET_RESTORE    ;PERFORM A MASTER RESET AND
                                                ; RESTORE MASTER REG TO NORMAL VALUE

                CALL    CLR_CG_PNT              ;CLEAR THE COUNTER GROUP POINTER.

                DO      5

                CALL    INC_CG_PNT              ;INCREMENT THE COUNTER GROUP POINTER.

                LXI     H,TEST25_DAT            ;SET EACH COUNTER TO COUNT UP.
                CALL    WRITE_MODE

                ENDDO
```

```
7BAD 2606
7BAD 2607 004D'CD                                  CALL    CLR_CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.
7BB0 2608
7BB0 2609 C5 46 00'00'21     ******************    DO      5
          77 05 3E           *
7BB8 2610                    *
7BB8 2611 0052'CD            *                     CALL    INC_CG_PNT      ;INCREMENT THE COUNTER GROUP POINTER.
7BBB 2612                    *
7BBB 2613 000E'2A            *                     LHLD    ZERO_DAT        ;MOVE THE FIRST DATA PATTERN TO BE
7BBE 2614 0025'22            *                     SHLD    TEMP_DAT        ; SENT TO THE COUNTER TO TEMP_DAT.
7BC1 2615                    *
7BC1 2616 C5 46 00'00'21     * ****************    DO      16
          77 10 3E           * *
7BC9 2617                    * *
7BC9 2618 0025'2A            * *                   LHLD    TEMP_DAT        ;MOVE A DATA PATTERN TO EXP_DAT...
7BCC 2619 0028'22            * *                   SHLD    EXP_DAT
7BCF 2620                    * *
7BCF 2621 00'28'21           * *                   LXI     H,EXP_DAT       ;...AND INCREMENT IT.
7BD2 2622 0000'CD            * *                   CALL    INC_WORD
7BD5 2623                    * *
7BD5 2624 00'25'21           * *             1$:   LXI     H,TEMP_DAT      ;WRITE A DATA PATTERN TO THE LOAD
7BD8 2625 0085'CD            * *                   CALL    WRITE_LOAD      ; REGISTER.
7BDB 2626                    * *
7BDB 2627 0105'CD            * *                   CALL    DISARM_CNTR     ;DISARM THE COUNTER.
7BDE 2628                    * *
7BDE 2629 011A'CD            * *                   CALL    LOAD_CNTR       ;LOAD THE COUNTER FROM THE LOAD REG.
7BE1 2630                    * *
7BE1 2631 0156'CD            * *                   CALL    INC_CNTR        ;INCREMENT THE COUNTER.
7BE4 2632                    * *
7BE4 2633 012F'CD            * *                   CALL    SAVE_CNTR       ;SAVE THE COUNTER IN THE HOLD REG.
7BE7 2634                    * *
7BE7 2635 00A9'CD            * *                   CALL    READ_HOLD       ;READ THE HOLD REGISTER.
7BEA 2636                    * *
7BEA 2637 00'28'21           * * **************    IFNEQ   EXP_DAT,REC_DAT,2       ;COMPARE THE EXPECTED AND
          02 0E 00'2B'11     * * *
          0BFB'CA 0000'CD    * * *
7BF8 2638                    * * *                                                 ; RECEIVED DATA.
7BF8 2639                    * * *
7BF8 2640 01CE'CD            * * *                 CALL    TEST21_ERROR    ;PRINT AN ERROR MESSAGE.
7BFB 2641                    * * *
7BFB 2642                    * * **************    ENDIF
7BFB 2643                    * *
7BFB 2644 0000'3A            * *                   LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7BFE 2645 0F                 * *                   RRC
7BFF 2646 0BD5'DA            * *                   JC      1$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7C02 2647                    * *
7C02 2648 37                 * *                   STC                     ;SET THE CARRY.
7C03 2649                    * *
7C03 2650 00'24'21           * *                   LXI     H,TEMP_SHIFT    ;MAKE THE NEXT DATA PATTERN TO BE SENT
7C06 2651 0000'22            * *                   SHLD    SHIFT_PNT       ; TO THE COUNTER BY SHIFTING A ONE
7C09 2652 0000'CD            * *                   CALL    SHIFTER         ; INTO THE PRESENT PATTERN.
7C0C 2653                    * *
7C0C 2654 002D'CD            * *                   CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7C0F 2655                    * *
7C0F 2656 35 00'00'21        * ****************    ENDDO
          70 C1 0BC9'C2      *
7C18 2657                    *
7C18 2658 35 00'00'21        ******************    ENDDO
          70 C1 0BB8'C2
7C21 2659
```

```
7C21 2660 3C D3
7C23 2661
7C23 2662
7C23 2663
7C23 2664
7C23 2665
7C23 2666
7C23 2667
7C23 2668
7C23 2669
7C23 2670
7C23 2671
7C23 2672
7C23 2673
7C23 2674
7C23 2675
7C23 2676
7C23 2677
7C23 2678
7C23 2679
7C23 2680
7C23 2681
7C23 2682
7C23 2683
7C23 2684
7C23 2685
7C23 2686
7C23 2687
7C23 2688
7C23 2689
7C23 2690
7C23 2691
7C23 2692
7C23 2693
7C23 2694
7C23 2695
7C23 2696
7C23 2697
7C23 2698
7C23 2699
7C23 2700
7C23 2701
7C23 2702
7C23 2703
7C23 2704
7C23 2705 0000'CD 26 3E
          0F 0000'3A
          C9 0C30'D2
          0F 0000'3A
          3D D3 0DB0'DA
7C39 2706
7C39 2707 003D'CD
7C3C 2708
7C3C 2709 00'20'21
7C3F 2710 0057'CD
7C42 2711
7C42 2712 004D'CD
7C45 2713
7C45 2714 C5 46 00'00'21      ******************
          77 02 3E            *
```

```
;*****************************************************************************
;
;       TEST26                  COMPARATOR TEST
;       ------                  ---------------
;
;               THIS TEST LOADS THE ALARM REGISTER WITH DATA PATTERNS AND
;       THE COUNTER WITH THE SAME PATTERN DECREMENTED BY ONE. COUNTERS 1 AND
;       2 COMPARE ENABLE ARE SET AND THE COUNTERS ARE INCREMENTED. THE COUNTER
;       OUTPUTS ARE THEN CHECKED TO MAKE SURE THAT AN ACTIVE HIGH OUTPUT HAS
;       OCCURED. THE DATA PATTERNS CONSIST OF ALL ONES, A ONE SHIFTED THROUGH
;       A ZERO FIELD AND A ZERO SHIFTED THROUGH A ONE FIELD.
;
;       TEST STEPS:
;
;               1) PERFORM MASTER RESET.
;               2) ENABLE COMPARATORS 1 AND 2.
;               3) LOAD ALARM REGISTERS 1 AND 2 WITH A DATA PATTERN.
;               4) LOAD COUNTERS 1 AND 2 WITH DECREMENTED VALUE OF THE DATA
;                  PATTERN.
;               5) INCREMENT COUNTERS ONE AND TWO.
;               6) READ THE STATUS REGISTER (SHOULD BE HEX C7).
;               7) PRINT ERROR IF ANY.
;               8) REPEAT STEPS 3 - 7 FOR ALL DATA PATTERNS.
;
;       ASSUMPTIONS: PREVIOUS INTERVAL TIMER TESTS HAVE RUN SUCCESSFULLY.
;
;       ERROR1: COUNTERS 1 AND/OR 2 OUTPUT BITS NOT SET.
;
;       NOTE: ADDITIONAL DATA CONTAINS DATA SENT TO COMPARATOR.
;
;       DEBUG: SUSPECT INTERVAL TIMER CHIP.
;
;*****************************************************************************

TEST26:         HEAD    <^X26>


                CALL    MASTER_RESET    ;PERFORM A MASTER RESET.

                LXI     H,TEST26_DAT    ;SET COUNTERS 1 AND 2 TO COMPARE
                CALL    WRITE_MASTER    ; WITH ALARM REGISTER MODE.

                CALL    CLR_CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.

                DO      2
```

```
7C4D 2715                    *
7C4D 2716 0052'CD            *                    CALL    INC_CG_PNT       ;INCREMENT THE COUNTER GROUP POINTER.
7C50 2717                    *
7C50 2718 00'22'21           *                    LXI     H,TEST26_DAT+2   ;INITIALIZE THE COUNTER MODE REG.
7C53 2719 006D'CD            *                    CALL    WRITE_MODE
7C56 2720                    *
7C56 2721 35 00'00'21        ******************   ENDDO
          70 C1 0C4D'C2
7C5F 2722
7C5F 2723 06 3E                                   MVI     A,<^X06>         ;LOAD DATA WITH COUNTER OUPUT BITS 1
7C61 2724 0028'32                                 STA     EXP_DAT          ; AND 2 SET INTO EXP_DAT.
7C64 2725
7C64 2726 004D'CD                          1$:    CALL    CLR_CG_PNT       ;CLEAR THE COUNTER GROUP POINTER.
7C67 2727
7C67 2728 C5 46 00'00'21     ******************   DO      2
          77 02 3E           *
7C6F 2729                    *
7C6F 2730 0052'CD            *                    CALL    INC_CG_PNT       ;INCREMENT THE COUNTER GROUP POINTER.
7C72 2731                    *
7C72 2732 0011'2A            *                    LHLD    ONE_DAT          ;LOAD ALL ONES INTO TEMP_DAT AND
7C75 2733 0025'22            *                    SHLD    TEMP_DAT         ; TIMER_DAT.
7C78 2734 0009'22            *                    SHLD    TIMER_DAT
7C7B 2735                    *
7C7B 2736 00'09'21           *                    LXI     H,TIMER_DAT      ;WRITE ALL ONES TO THE ALARM REG.
7C7E 2737 00B5'CD            *                    CALL    WRITE_ALARM
7C81 2738                    *
7C81 2739 00'09'21           *                    LXI     H,TIMER_DAT      ;DECREMENT THE DATA PATTERN...
7C84 2740 0000'CD            *                    CALL    DECR_WORD
7C87 2741                    *
7C87 2742 00'09'21           *                    LXI     H,TIMER_DAT      ;...AND WRITE IT TO THE LOAD REG.
7C8A 2743 0085'CD            *                    CALL    WRITE_LOAD
7C8D 2744                    *
7C8D 2745 011A'CD            *                    CALL    LOAD_CNTR        ;LOAD THE COUNTER FROM THE LOAD REG.
7C90 2746                    *
7C90 2747 0156'CD            *                    CALL    INC_CNTR         ;INCREMENT THE COUNTER.
7C93 2748                    *
7C93 2749 35 00'00'21        ******************   ENDDO
          70 C1 0C6F'C2
7C9C 2750
7C9C 2751 1D DB                                   IN      ITCSR            ;READ THE STATUS REG.
7C9E 2752 06 E6                                   ANI     <^X06>           ;MASK OUT BITS 1 AND 3-7.
7CA0 2753 002B'32                                 STA     REC_DAT          ;STORE IN REC_DAT.
7CA3 2754
7CA3 2755 00'28'21           ******************   IFNEQ   EXP_DAT,REC_DAT,1        ;COMPARE THE EXPECTED AND
          01 0E 00'2B'11     *
          0CB4'CA 0000'CD    *
7CB1 2756                    *                                                     ; RECEIVED DATA.
7CB1 2757                    *
7CB1 2758 026E'CD            *                    CALL    TEST26_ERROR     ;PRINT AN ERROR MESSAGE.
7CB4 2759                    *
7CB4 2760                    ******************   ENDIF
7CB4 2761
7CB4 2762 0000'3A                                 LDA     ERROR_CON        ;IF THE ERROR LOOPING FLAG IS SET...
7CB7 2763 0F                                      RRC
7CB8 2764 0C64'DA                                 JC      1$               ;...JUMP TO BEGINNING OF ERROR LOOP.
7CBB 2765
7CBB 2766
7CBB 2767 000F'2A                                 LHLD    ONE_SHIFT        ;LOAD PATTERN OF ONE SHIFTED THROUGH
7CBE 2768 0025'22                                 SHLD    TEMP_DAT         ; ZERO FIELD INTO TEMP_DAT.
7CC1 2769
```

```
7CC1 2770 C5 46 00'00'21      ******************          DO      16
          77 10 3E            *
7CC9 2771                     *
7CC9 2772 002D'CD             *                           CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7CCC 2773                     *
7CCC 2774 004D'CD             *                   2$:     CALL    CLR_CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.
7CCF 2775                     *
7CCF 2776 C5 46 00'00'21      * ****************          DO      2
          77 02 3E            * *
7CD7 2777                     * *
7CD7 2778 0025'2A             * *                         LHLD    TEMP_DAT        ;LOAD TEMP_DAT INTO TIMER_DAT.
7CDA 2779 0009'22             * *                         SHLD    TIMER_DAT
7CDD 2780                     * *
7CDD 2781 0052'CD             * *                         CALL    INC_CG_PNT      ;INCREMENT THE COUNTER GROUP POINTER.
7CE0 2782                     * *
7CE0 2783 00'09'21            * *                         LXI     H,TIMER_DAT     ;WRITE THE DATA TO THE ALARM REG.
7CE3 2784 00B5'CD             * *                         CALL    WRITE_ALARM
7CE6 2785                     * *
7CE6 2786 00'09'21            * *                         LXI     H,TIMER_DAT     ;DECREMENT THE DATA PATTERN....
7CE9 2787 0000'CD             * *                         CALL    DECR_WORD
7CEC 2788                     * *
7CEC 2789 00'09'21            * *                         LXI     H,TIMER_DAT     ;...AND WRITE IT TO THE LOAD REG.
7CEF 2790 0085'CD             * *                         CALL    WRITE_LOAD
7CF2 2791                     * *
7CF2 2792 011A'CD             * *                         CALL    LOAD_CNTR       ;LOAD THE COUNTER FROM THE LOAD REG.
7CF5 2793                     * *
7CF5 2794 0156'CD             * *                         CALL    INC_CNTR        ;INCREMENT THE COUNTER.
7CF8 2795                     * *
7CF8 2796 35 00'00'21         * ****************          ENDDO
          70 C1 0CD7'C2       *
7D01 2797                     *
7D01 2798 1D DB               *                           IN      ITCSR           ;READ THE STATUS REGISTER.
7D03 2799 06 E6               *                           ANI     <^X06>          ;MASK OUT BITS 0 AND 3-7.
7D05 2800 002B'32             *                           STA     REC_DAT         ;STORE IN REC_DAT.
7D08 2801                     *
7D08 2802 00'28'21            * ****************          IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'2B'11      * *
          0D19'CA 0000'CD     * *
7D16 2803                     * *                                                         ; RECEIVED DATA,
7D16 2804                     * *
7D16 2805 026E'CD             * *                         CALL    TEST26_ERROR    ;PRINT AN ERROR MESSAGE.
7D19 2806                     * *
7D19 2807                     * ****************          ENDIF
7D19 2808                     *
7D19 2809 0000'3A             *                           LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7D1C 2810 0F                  *                           RRC
7D1D 2811 0CCC'DA             *                           JC      2$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7D20 2812                     *
7D20 2813 B7                  *                           ORA     A               ;CLEAR THE CARRY TO SHIFT THE ONE
7D21 2814 00'24'21            *                           LXI     H,TEMP_SHIFT    ; THROUGH THE ZERO FIELD.
7D24 2815 0000'22             *                           SHLD    SHIFT_PNT
7D27 2816 0000'CD             *                           CALL    SHIFTER
7D2A 2817                     *
7D2A 2818 35 00'00'21         ******************          ENDDO
          70 C1 0CC9'C2
7D33 2819
7D33 2820
7D33 2821 0012'2A                                         LHLD    ZERO_SHIFT      ;LOAD A ZERO IN A ONE FIELD INTO
7D36 2822 0025'22                                         SHLD    TEMP_DAT        ; TEMP_DAT.
7D39 2823
```

```
7D39 2824 C5 46 00'00'21      ******************          DO      16
          77 10 3E            *
7D41 2825                     *
7D41 2826 002D'CD             *                           CALL    CHECK_RD        ;CALL BOOT CODE IF UNDER RD CONTROL
7D44 2827                     *
7D44 2828 004D'CD             *                   3$:     CALL    CLR_CG_PNT      ;CLEAR THE COUNTER GROUP POINTER.
7D47 2829                     *
7D47 2830 C5 46 00'00'21      * ****************          DO      2
          77 02 3E            * *
7D4F 2831                     * *
7D4F 2832 0025'2A             * *                         LHLD    TEMP_DAT        ;LOAD TEMP_DAT INTO TIMER_DAT.
7D52 2833 0009'22             * *                         SHLD    TIMER_DAT
7D55 2834                     * *
7D55 2835 0052'CD             * *                         CALL    INC_CG_PNT      ;INCREMENT THE COUNTER GROUP POINTER.
7D58 2836                     * *
7D58 2837 014D'CD             * *                         CALL    CLR_OUTPUT      ;CLEAR THE COUNTER OUTPUT BIT.
7D5B 2838                     * *
7D5B 2839 00'09'21            * *                         LXI     H,TIMER_DAT     ;WRITE THE PATTERN TO THE ALARM REG.
7D5E 2840 00B5'CD             * *                         CALL    WRITE_ALARM
7D61 2841                     * *
7D61 2842 00'09'21            * *                         LXI     H,TIMER_DAT     ;DECREMENT THE DATA PATTERN...
7D64 2843 0000'CD             * *                         CALL    DECR_WORD
7D67 2844                     * *
7D67 2845 00'09'21            * *                         LXI     H,TIMER_DAT     ;...AND WRITE IT TO THE LOAD REG.
7D6A 2846 0085'CD             * *                         CALL    WRITE_LOAD
7D6D 2847                     * *
7D6D 2848 011A'CD             * *                         CALL    LOAD_CNTR       ;LOAD THE COUNTER FROM THE LOAD REG.
7D70 2849                     * *
7D70 2850 0156'CD             * *                         CALL    INC_CNTR        ;INCREMENT THE COUNTER.
7D73 2851                     * *
7D73 2852 35 00'00'21         * ****************          ENDDO
          70 C1 0D4F'C2       *
7D7C 2853                     *
7D7C 2854 1D DB               *                           IN      ITCSR           ;READ THE STATUS REGISTER.
7D7E 2855 06 E6               *                           ANI     <^X06>          ;MASK OUT BITS 0 AND 3-7.
7D80 2856 002B'32             *                           STA     REC_DAT         ;STORE IN REC_DAT.
7D83 2857                     *
7D83 2858 00'28'21            * ****************          IFNEQ   EXP_DAT,REC_DAT,1       ;COMPARE THE EXPECTED AND
          01 0E 00'2B'11      * *
          0D94'CA 0000'CD     * *
7D91 2859                     * *                                                         ; RECEIVED DATA.
7D91 2860                     * *
7D91 2861 026E'CD             * *                         CALL    TEST26_ERROR    ;PRINT AN ERROR MESSAGE.
7D94 2862                     * *
7D94 2863                     * ****************          ENDIF
7D94 2864                     *
7D94 2865 0000'3A             *                           LDA     ERROR_CON       ;IF THE ERROR LOOPING FLAG IS SET...
7D97 2866 0F                  *                           RRC
7D98 2867 0D44'DA             *                           JC      3$              ;...JUMP TO BEGINNING OF ERROR LOOP.
7D9B 2868                     *
7D9B 2869 37                  *                           STC                     ;SET THE CARRY TO SHIFT THE ZERO
7D9C 2870 00'24'21            *                           LXI     H,TEMP_SHIFT    ; THROUGH THE ONE FIELD.
7D9F 2871 0000'22             *                           SHLD    SHIFT_PNT
7DA2 2872 0000'CD             *                           CALL    SHIFTER
7DA5 2873                     *
7DA5 2874 35 00'00'21         ******************          ENDDO
          70 C1 0D41'C2
7DAE 2875
7DAE 2876 3C D3                                           TAIL
7DB0 2877
```

```
7DB0 2878
7DB0 2879                    ;*****************************************************************************
7DB0 2880                    ;
7DB0 2881                    ;  LAST TEST FOR EVERY CONSOL_SECTION....FORCES END OF SECTION
7DB0 2882                    ;
7DB0 2883                    ;*****************************************************************************
7DB0 2884
7DB0 2885 0000'CD 51 3E      END_TEST:       HEAD    <^X51>
          0F 0000'3A
          C9 0DBD'D2
          0F 0000'3A
          3D D3 0DC8'DA
7DC6 2886
7DC6 2887 3C D3                              TAIL
7DC8 2888
```

```
$PSW                            = 00000006
$SP                             = 00000006
A                               = 00000007
ALARM_REG1                      = 0C000007
ALARM_REG2                      = 0000000F
ALLOWP                          = 00000003
APTFLG                          = 00000004
AUTO                            = 00000005
B                               = 00000000
BOOTEN                          = 00000007
BOOTF                           = 00000001
C                               = 00000001
CG1_HOLD_H                      = 00000019
CG1_MODE_E                      = 00000001
CG_PNT                            00000007 R     02
CHECK_RD                          0000002D R     02
CLRACK                          = 000000A9
CLRACL                          = 00000033
CLRATN                          = 000000AB
CLRBSY                          = 0000002E
CLRCLK                          = 00000020
CLRCSK                          = 00000024
CLRCSR                          = 00000038
CLRDCL                          = 00000031
CLRHLT                          = 000000AF
CLRLIT                          = 0000003C
CLRMCI                          = 00000029
CLRMCK                          = 0000002B
CLRPFI                          = 000000AD
CLRSS                           = 00000022
CLRTIM                          = 000000A5
CLRTMR                          = 0000002C
CLRUPC                          = 000000A8
CLRUPK                          = 00000026
CLRWCK                          = 00000036
CLR_CG_PNT                        0000004D R     02
CLR_OUTPUT                        0000014D R     02
CLR_OUTPUT_DAT                  = 000000E0
CNT                             = 0000082D
CNTA                            = 00000FA0
CNTLC_TYPED                       ********   X   02
COMPARE                           ********   X   02
CON_ADD_DAT                       ********   X   02
CON_BEGIN_TEST                    ********   X   02
CON_ERROR                         ********   X   02
CON_ERR_NUM                       ********   X   02
CON_EXP_DAT                       ********   X   02
CON_MOD_DAT                       ********   X   02
CON_MSK_DAT                       ********   X   02
CON_REC_DAT                       ********   X   02
CPATTN                          = 00000083
CPUACK                          = 00000082
CSR07                           = 00000085
CSR15                           = 00000086
CSR23                           = 00000087
CYCLE_CNT                         0000000B R     02
D                               = 00000002
DCLO                            = 00000002
DECR_WORD                         ********   X   02
DISARM_CNTR                       00000105 R     02
DISARM_CNTR_DAT                 = 000000C0
DISMEM                          = 000000A7
DISPAR                          = 000000A1
DIS_DP_SEQ                      = 000000E8
DOLOOP                            ********   X   02
E                               = 00000003
ENBMEM                          = 000000A6
ENBPAR                          = 000000A0
END_TEST                          00000DB0 R     02
ENTRY                             00000000 R     02
EOS_FLG                           ********   X   02
ERROR_CON                         ********   X   02
EXP_DAT                           00000028 R     02
EXP_SHIFT                         00000027 R     02
FILE_NAME                         00000005 R     02
GCVECT                            ********   X   02
H                               = 00000004
HEAD                            = 00000000
HLTFLG                          = 00000002
HOLD_REG                        = 00000010
INC_CG_PNT                        00000052 R     02
INC_CNTR                          00000156 R     02
INC_CNTR_DAT                    = 000000F0
INC_CYCLE_CNT                     00000100 R     02
INC_WORD                          ********   X   02
INITH                           = 00000035
INITL                           = 00000034
ITCSR                           = 0000001D
ITDB                            = 0000001C
L                               = 00000005
LITERAL                         = 00000001
LOAD_CNTR                         0000011A R     02
LOAD_CNTR_DAT                   = 00000040
LOAD_MOD_DAT                      0000015F R     02
LOAD_REG                        = 00000008
LVL                             = 00000000
M                               = 00000006
MACSTK1                         = 0000082D
MACSTK2                         = 00000829
MACSTK3                         = 0000082B
MACSTK4                         = 00000820
MASTER_MODE                     = 00000017
MASTER_RESET                      0000003D R     02
MASTER_RESET_RESTORE              00000042 R     02
MIPFLG                          = 000040D9
MISC2                           = 00000001
MM_BITS                           000000ED R     02
MODE_REG                        = 00000000
MOVER                             ********   X   02
MWCS_A                            ********   X   02
NA_OTHER                          ********   X   02
NORMAL_MASTER                     0000000C R     02
OKV15                           = 00000007
ONE_DAT                           00000011 R     02
```

```
ONE_SHIFT                 0000000F R    02
PARITY_VECTOR           = 00004C22
POWER_VECTOR            = 00004C25
READ                    = 00000080
READJ1                  = 00000003
READJ2                  = 00000004
READ_ALARM                000000C7 R    02
READ_HOLD                 000000A9 R    02
READ_ITDB                 000000E2 R    02
READ_LOAD                 00000091 R    02
READ_MASTER               0000005F R    02
READ_MODE                 00000079 R    02
REC_DAT                   0000002B R    02
REC_SHIFT                 0000002A R    02
REMDIS                  = 00000006
RESET_DATA              = 000000FF
SAVE_CNTR                 0000012F R    02
SAVE_CNTR_DAT           = 000000A0
SETACK                  = 000000A8
SETACL                  = 00000032
SETATN                  = 000000AA
SETBSY                  = 0000002F
SETCLK                  = 00000021
SETCSK                  = 00000025
SETCSR                  = 00000039
SETDCL                  = 00000030
SETHLT                  = 000000AE
SETLIT                  = 0000003D
SETMCI                  = 00000028
SETMCK                  = 0000002A
SETPFI                  = 000000AC
SETSS                   = 00000023
SETTIM                  = 000000A4
SETTMR                  = 0000002D
SETUPC                  = 000000A9
SETUPK                  = 00000027
SETWCK                  = 00000037
SET_FOR_CONT              ********   X   02
SET_OUTPUT                00000144 R    02
SET_OUTPUT_DAT          = 000000E8
SHIFTER                   ********   X   02
SHIFT_CNT                 00000008 R    02
SHIFT_PNT                 ********   X   02
SKIP_TEST                 ********   X   02
SYM                     = 0000082D
TEMP_DAT                  00000025 R    02
TEMP_SHIFT                00000024 R    02
TEST1E                    0000031A R    02
TEST1E_DAT                00000014 R    02
TEST1E_ERROR              00000168 R    02
TEST1F                    00000426 R    02
TEST1F_ERROR              000001A2 R    02
TEST20                    0000052E R    02
TEST20_ONES_ZEROS         000002A7 R    02
TEST21                    000006AA R    02
TEST21_ERROR              000001CE R    02
TEST22                    000007BD R    02
TEST22_DAT                00000016 R    02
TEST22_ERROR              0000020D R    02
TEST23                    000009A2 R    02
TEST23_DAT                00000018 R    02
TEST24                    00000AAE R    02
TEST24_DAT                0000001C R    02
TEST24_ERROR              00000247 R    02
TEST25                    00000B77 R    02
TEST25_DAT                0000001E R    02
TEST26                    00000C23 R    02
TEST26_DAT                00000020 R    02
TEST26_ERROR              0000026E R    02
TIMER_DAT                 00000009 R    02
TTCR                    = 0000004B
TTDB                    = 00000048
TTMODE                  = 0000004A
TTSR                    = 00000049
TUCR                    = 00000047
TUDB                    = 00000044
TUMODE                  = 00000046
TUSR                    = 00000045
UPC14                   = 00000084
UPC14A                  = 00000000
VERSION                   00000003 R    02
VNORML                  = 000000A3
VSHIFT                  = 000000A2
WCSB0                   = 00000008
WCSB1                   = 00000009
WCSB2                   = 0000000A
WRITE                   = 000000CC
WRITE_ALARM               000000B5 R    02
WRITE_HOLD                0000009D R    02
WRITE_ITDB                000000D9 R    02
WRITE_LOAD                00000085 R    02
WRITE_MASTER              00000057 R    02
WRITE_MODE                0000006D R    02
Y                       = 00000044
YBUSRD                  = 000000EC
ZERO_DAT                  0000000E R    02
ZERO_SHIFT                00000012 R    02
```

```
                                        +----------------+
                                        ! Psect synopsis !
                                        +----------------+
PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
. BLANK .                       00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
CPU4                            00000DC8  ( 3528.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE

                                      +-------------------------+
                                      ! Performance indicators !
                                      +-------------------------+
Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     23    00:00:00.05    00:00:00.28
Command processing                 34    00:00:00.26    00:00:00.76
Pass 1                           1355    00:00:32.44    00:00:34.41
Symbol table sort                   0    00:00:00.30    00:00:00.33
Pass 2                           1409    00:00:12.70    00:00:14.70
Symbol table output                29    00:00:00.16    00:00:00.18
Psect synopsis output               5    00:00:00.02    00:00:00.02
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             2859    00:00:45.93    00:00:50.68

The working set limit was 1000 pages.
375846 bytes (735 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 211 non-local and 131 local symbols.
4912 source lines were read in Pass 1, producing 44 object records in Pass 2.
151 pages of virtual memory were used to define 150 macros.

                                         +---------------------------+
                                         ! Macro library statistics !
                                         +---------------------------+
Macro library name                        Macros defined
------------------                        --------------
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1                    0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

/SHOW=(BIN)/LIST=ENKBD.LIS/OBJECT=ENKBD.OBJ 8085MAC.MAR+MACDEF.MAC+[]ENKBD.MAC
```